Exploring the Depths of Sound: An Analysis of CNNs for Environmental Sound Classification

George P. Prodan[†], Elaheh Ahmadieslamloo[‡]

***Abstract*—The field of Environment Sound Classification (ESC) focuses on identifying and categorizing sounds in various surroundings. Convolutional Neural Networks (CNNs) have become a powerful tool for ESC due to their ability to learn and extract features from audio data. In this project, we explore different CNN architectures using traditional Conv-Pool layers, residual layers, and Convolutional Recurrent Neural Networks (CRNNs) for ESC. The models are built with up to 6 layers, and we experiment with various regularization techniques to optimize their performance. Our results provide a comprehensive analysis of different approaches to ESC, including width scaling, depth scaling, and the use of pooling layers. Additionally, we perform a grid search to fine-tune the hyperparameters of our models. The grid search takes into account the dropout probability, augmentation probability, and the augmentation methods used.**

***Index Terms*—Environmental Sound Classification, Convolutional Neural Networks, Residual Networks, Recurrent Neural Networks, Optimization**

I. Introduction

The field of Environment Sound Classification (ESC) concentrates on identifying and categorizing sounds in various surroundings. With the rise of audio-based technologies such as smart speakers, virtual assistants, and surveillance systems, the creation of efficient algorithms for ESC is becoming a crucial issue. The dynamic and unstructured nature of acoustic environments poses a significant practical challenge for designing suitable features for ESC. Many current ESC methods design features based on prior knowledge of acoustic environments and then train a classifier using these features to determine the category probability of each environmental sound signal [1].

A variety of signal processing or machine learning algorithms have been developed to the tackle this problem, including matrix factorization [2], dictionary learning [3], wavelet filterbanks unsupervised learning [4], and most recently deep neural networks [5]. The availability of new data in the context of environmental sound classification and the development of numerous data augmentation techniques have been followed by an increase of deep learning models in this field.

Convolutional Neural Networks (CNNs) have emerged as a powerful tool for environment sound classification due to their ability to learn and extract features from raw audio data. CNNs and other CNN-based models have been successfully applied to various audio classification tasks, including speech recognition [6], music genre classification [7], and environmental sound classification [8].

In this project, we start from implementing a basic CNN architecture inspired from the one we have seen during the second laboratory of the course of Human Data Analytics and we explore various scaling techniques. First, we perform width scaling, followed by depth scaling. Then, we incorporate residual layers and compare their performance with the traditional architecture. Additionally, we also experiment with Convolutional Recurrent Neural Network (CRNN) with and without residual layers. To further optimize the models, we also explore various regularization techniques and present the results of our experiments. Our contribution aims to provide a comprehensive analysis of different approaches to ESC by exploring the performances, computational costs and possible optimizations.

II. Related Work

The use of Convolutional Neural Networks (CNNs) has been widely adopted in environment sound classification due to their ability to learn and extract features from raw audio data. Several studies have demonstrated the effectiveness of CNNs in this field. Lee et al. (2017) proposed a multi-scale Convolutional Neural Network (CNN) architecture for environment sound classification and achieved an accuracy of 63.4% on the DCASE2017 dataset. The authors demonstrated that their multi-scale CNN architecture can effectively capture different aspects of the sound signals and achieve improved performance compared to traditional single-scale CNN architectures. This study highlights the importance of considering multi-scale representations in environment sound classification tasks and provides a valuable contribution to the field. [9].Piczak (2015) used a temporal convolutional network with an attention mechanism to classify environment sounds, resulting in an accuracy of 72.2% on the ESC-10 dataset. The author showed that the combination of temporal CNNs and an attention mechanism can effectively capture the temporal dynamics of the sound signals and achieve improved performance compared to traditional feedforward neural networks. This study is one of the earliest works that applied deep learning techniques to environment sound classification and provides valuable insights into the potential of using temporal CNNs with attention mechanisms for this task [10].Abdoli et al proposed an end-to-end approach for environmental sound classification based on a 1D Convolution Neural Network (CNN) that learns a representation directly from the audio signal. performance of the proposed end-to-end

[†]MSc in Physics of Data, University of Padova, email: {georgepantelimon.prodan}@studenti.unipd.it

[‡]MSc in Physics of Data, University of Padova, email: {elaheh.ahmadieslamloo}@studenti.unipd.it

approach in classifying environmental sounds was assessed on the UrbanSound8k dataset and they got %89 of accuracy. [11]Zhang et al used Convolutional Neural Network-Gated Recurrent Unit for classifying environment sound on different dataset and they achieved good classification accuracy for the three datasets, ESC-10 (92.30%), ESC-50 (87.43%), and UrbanSound8K (96.10%) [12]

A. *Residual Layers*

The ResNet architecture introduced by Zhang et al [13] revolutionized the field of CNNs by effectively addressing the issue of "gradient vanishing" in deep networks with numerous stacked layers. This design marked a major milestone in deep architecture for image processing tasks. However, this paper only focuses on image processing and does not address other problems such as the one presented in this project.

B. *Recurrent Convolutional Neural Networks (RCNNs)*

Recurrent Convolutional Neural Networks (RCNNs) are a type of deep learning architecture that combines the features of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). RCNNs use convolutional layers to extract features from input data, such as images or signals, and then pass these features to a recurrent layer to model temporal dependencies in the data [14].

As far as we know, RCNNs have been proposed for extracting features from raw waveforms [15] using LSTMs. However, in this project we plan to use this type of architecture in learning the features from MFCCs instead, using GRUs.

C. *Data Augmentation*

When training a neural network, it is common to use techniques such as data augmentation to enhance the performance of the model. Data augmentation involves altering the existing training data in some way, such as rotating or flipping images, to create new variations of the data. By augmenting the data during the training process, the model can learn to recognize patterns and features in the data that may not have been captured in the original data set, making it more robust and less prone to overfitting.

There are a lot of data augmentation options for audio signals. In [16] there are proposed approaches such as speed change, pitch shifting, volume gain, inserting Gaussian noise, time shifting, wow resampling, clipping, or harmonic distortion. However, there are also techniques that operate directly on 2D representations, similar to traditional image augmentation methods like cropping, zooming, and masking [17].

III. Processing Pipeline

Our implementation is based on TensorFlow/Keras library of Python [18]. We develop our model by creating a subclass of the *tf.keras.Model*. In this way, we can easily run width and depth scaling or substitute the traditional convolutional layers with the the residual ones.

1) *Data Pre-processing* - We pre-process and store the audio clips at a sample rate of 44100Hz and a single channel.
2) *Data Loading* - We define a subclass of *tf.keras.Sequence* to iterate over the dataset. In this way we can augment the data randomly during the training. In live data augmentation, the data is transformed on the fly as it is being fed into the network, rather than being pre-processed and stored in a separate data set. This approach allows for a larger and more diverse data set to be used for training, which can lead to better results. However, depending on the dataset it can also be computationally expensive, as the transformations must be performed in real-time.
3) *Model Training* - We run the model for 40 - 100 epochs. After every iteration of training, the validation and training errors are monitored to detect any overfitting or underfitting. If we notice overfitting we can apply regularization methods such as dropout layers or we may increase data augmentation probability to enhance the model's ability to generalize. We run all the experiments on Colab Pro. In this case, one epoch takes between 1 to 3 seconds, depending on the size of the model.
4) *Model Testing* - We test the model on 10% of the dataset as explained in the next section.

All these steps are repeated to run multiple experiments such as width or depth scaling, or optimization experiments.

Width scaling is a method of scaling the width (i.e. the number of filters) of a convolutional layer in a convolutional neural network (CNN). It is used to control the model capacity (i.e. the number of parameters) and prevent overfitting. To implement width scaling in a convolutional neural network, you can simply scale the number of filters in each convolutional layer. For example, if you have a layer with 32 filters, you could scale the width by a factor of 2, to 64 filters.

We start by creating different CNN models of different widths. We obtain them by increasing the number of filters, F, from 8 to 256. Further exploration takes place by adding extra blocks (depth scaling), from 2 to 6 blocks, or using different types of blocks (e.g. with residual layers). We conclude our first part of the experiments by investigating RCNNs with GRU.

In the second part of the project, we try different optimization techniques including data augmenatation and dropout layers.

IV. Signals and Features

For training our network, we extracted some important spectral features from each audio frame. In this work, we focus on two important feature families extracted from the spectrogram that is obtained using the short time Fourier transform (STFT). Mel-spectrogram and MFCC (Mel-frequency Cepstral Coefficients) are two commonly used features in audio processing and analysis, especially for environmental sound classification.

A. Mel spectrogram

It represents the spectrogram in the Mel scale [19]. The formula for converting from frequency to Mel scale is

$$M(f) = 1125log(1 + f/700)$$

A total of 128 frequency bands have been considered. A Mel-spectrogram is a visual representation of the spectral content of an audio signal, where the frequency axis is transformed from linear to Mel scale. The Mel scale is designed to better match the human auditory system's perception of sound and is often used in speech and music processing.

B. MFCC

The Mel-Frequency Cepstral Coefficients (MFCCs) are other features extracted by applying a Discrete Cosine Transform (DCT) to the log-compressed Mel scale power spectrum [20]. MFCCs are a set of coefficients that capture the spectral envelope of a sound. They are derived from the Mel-spectrogram by applying a series of mathematical operations, including the discrete cosine transform (DCT). The resulting coefficients can be used as features to represent an audio signal for machine learning tasks, such as sound classification.

C. Dataset

We use the ESC-50 dataset, which consists of 50 classes of sounds from various environments. The split the dataset into training, validation, and test sets, is the common one: 80/10/10 split, meaning that 80% of the data is used for training, 10% for validation, and 10% for testing.

We prepare the dataset by performing a stratified split which ensures that the class distribution in the three sets is representative of the class distribution in the whole dataset. This means that each set contains roughly the same proportion of samples from each class, which is important to avoid introducing bias during the evaluations. To perform a stratified split, you would first group the samples by class, then randomly split each group into training, validation, and test sets so that the class distribution is maintaned in each set. By doing it in this way, we ensure that the model has enough examples of each class in the training set to learn from, and that the validation and test sets reflect the real-world distribution of classes.

V. Learning Framework

The basic CNN architecture we implement consists of two 2D convolutional and pooling layers, including batch normalization, rectified linear units (ReLU) as activation functions. It is followed by another extra pooling layer if needed and the output block that consists in two fully-connected layers of 150 and 50 hidden units. A dropout layer is applied after the last dense layer. The number of filters and the number of layers in the network can be specified as arguments when instantiating the model. The number of classes can also be specified. For ESC-50 we use have 50 classes. If the residual argument is set to True, the network will use residual connections in addition to the standard layers. In addition to traditional CNNs, the framework also allows for building Recurrent CNNs, which can be with or without residual connections. The simplified architecture versions of two layers are illustrated in Figure 1. As expected, the extra blocks are added right before the last pooling layer.

filters	test acc, %	param. (M)	best epoch / 50
4	86.00	0.5	50
8	86.50	1.0	35
16	90.50	2.0	25
32	88.50	4.0	29
64	89.50	8.0	35
128	90.00	16.2	23
256	87.00	33.1	18

TABLE 1: Width scaling results showing the test accuracy, number of parameters, best epoch out of 50 (with the best validation loss) as a function of the number of filters.

The number of filters doubles only after the first ConvPool block. The next blocks have the same number of filters, $2F$. For the RRCNN and RCNN we implement the GRU cell as it needs less computational requirements compared to LSTMs.

VI. Results

A. Initial Experiments

Here we experiment only on the traditional CNN architecture. We try different convolution or pooling kernels, and also different number of filters (width scaling). The initial experiments are performed with $F = 32$ filters and without the last pooling layer illustrated in Figure 1.

- **Filter Size** - we run the model on three different square filters of size 3,5 and 7. For the 5x5 filter the test accuracy drops from 88.5% to 84%. The larger kernel is slightly worse by 1%. Therefore, we decide to use only 3x3 kernels in the next experiments.
- **Pool Size** - larger pooling sizes can reduce more the spatial dimensions of the feature map, but may also lose important information. For a square pooling kernel of size 2, we got an accuracy of 88.5%. We observe that increasing the size to 4 and 8 decreases the test accuracy to 84% and 79%. Moreover, such small feature sizes will not allow us to add another pooling layers in the next blocks, so it will reduce the architecture complexity. Therefore, we consider 2x2 pooling kernel as the best option.
- **Width Scaling** - We sum up the results in Table 1. By monitoring the best epoch with the lowest validation cost, we observe that the model overfits the data when using more filters.

B. Depth scaling

We build on the two layer architecture presented in Fig. 1 by inserting additional layers up to 6. We try both traditional Conv-Pool layers and residual layers. For a fair comparison we use pooling layers such that the number of parameters for to be consistent. For instance, an extra max-pooling layer with kernel and stride of size (10, 1) is used for the tradional CNN

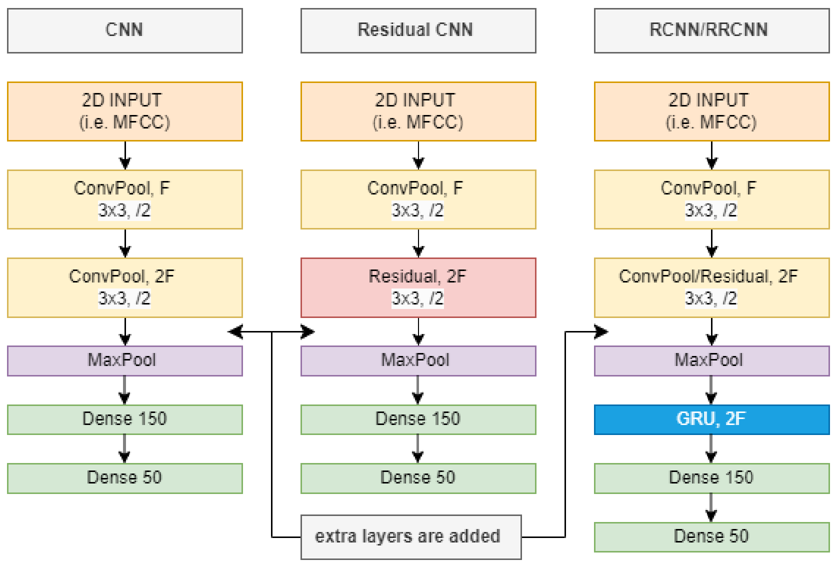

Fig. 1: The proposed architectures for running the experiments: CNN, CNN with residual layers, and recurrent CNN with or without residual layers. The extra blocks are added before the last pooling layer as illustrated in the figure.

with two layers to reduce the size of the features, such that the number of parameters is decreased from 16.2M to 2.3M. By adding another layer there is no need in performing an extra convolution as ConvPool already includes it. However, starting from layer 4 we deactive the pooling layer in the ConvPool block as we cannot reduce the feature size anymore. We repeat the procedure by using also a GRU unit after the convolutional part. In this case, GRU expects two dimensional inputs, so we apply pooling and reshape our features accordingly. By doing this procedure, the number of trainable parameter decreases by roughly two times.

C. *Regularization techniques*

There are two ways in which we attempt to optimize our models:

1) **by adding dropout layers** - we add one or two dropout layers with a probability between 0.3 and 0.5
2) **by augmentation** - we use frequency masks, meaning that we randomly mask out a frequency range on the MFCC spectrogram. Also, we implement time masking. Similar to frequency masks, except that we randomly block out ranges of time this time. Finally, we add Gaussian noise.

We perform a grid search of 60 epochs runs taking into account the dropout probability, augmentation probability and the augmentation methods we use. Because of computational reasons, we choose to perform the grid search on the residual CNN with 3 blocks, as we have noticed that the recurrent networks more time consuming.

Our grid search shows that the best choice would be a medium augmentation probability around 40%, a high dropout probability of 50%, applying both time and frequency masks simultaneously and regarding the maximum level of Gaussian noise, 4% is doing better than 2%. We do not state that these values are the best, because our grid search was limited on the time and the computational resources available.

D. *Summary of results*

In the Table 3 we present our final results we got on 100 epochs and using the hyper-parameters obtained from the previous grid search. We compare our models to SoundNet [21], a one dimensional CNN with 5 layers, implemented for this specific task but taking raw audio as an input. We implement this CNN and run it using the same set-up. We show the performance of a pre-trained transformer as well. We have not tested it, so the test accuracy is retrieved from the literature [22]. It is worth to mention that even our models do not outperform this baseline, the results are slightly close.

VII. Concluding Remarks

Our project describes a study on building deep learning models for the ESC task. The models were built on two-layer architectures and additional layers were added up to 6 layers. We implemented both traditional Conv-Pool layers and residual layers and used pooling layers to reduce the number of parameters while keeping the number of parameters consistent. In this way we were looking to optimize the results taking into account the computational resources involved in the process.

We performed a grid search to find a set that is closer to the optimal hyper-parameters. A further work could be running a larger grid search having a better resolution and including more hyper-parameters such as the batch size, learning rate, optimizer and, also the ones related to the architecture (i.e. number of filters, depth and so on). Another idea would be in experimenting with other RNN cells such as LSTM or trying different input representations.

We compared the performance of our models to SoundNet, a one-dimensional CNN with 5 layers and a pre-trained

# block	Block 2		Block 3		Block 4		Block 5		Block 6	
Residual	no	yes	no	yes	no	yes	no	yes	no	yes
CNN	88.5	86.50	90.00	**91.50**	**91.50**	87.50	90.50	89.00	91.00	89.00
# parameters	2.3M	3.0M	3.8M	5.6M	4.1M	6.8M	5.6M	8.0M	5.9M	9.2M
RCNN	**93.00**	91.50	92.00	92.00	88.50	**93.50**	92.00	92.00	89.00	92.00
# parameters	0.7M	1.3M	1.3M	2.7M	2.0M	3.9M	2.6M	5.0M	3.2M	6.2M

TABLE 2: Depth scaling results of the architectures we implemented. Test accuracy and the number of parameters are provided for each model. Best performance is highlighted for each architecture type.

Network	Input	Test Acc., %
Our best CNN	MFCC	91.5
Our best ResCNN	MFCC	91.5
Our best RCNN	MFCC	93.0
Our best RRCNN	MFCC	93.5
SoundNet [21]	Raw	87.5
Pre-trained transformer* [22]	Mel	95.7

TABLE 3: Summary of the results. (*) The pre-trained transformer has not been implemented by us and the corresponding accuracy is taken from literature.

transformer. The final results showed that our models were close to the performance of the pre-trained transformer, but did not outperform it.

Working on this project helped us understand how to build CNNs from scratch and how can we optimize them for a specific task. We did this using *tensorflow/keras* and we consolidated the knowledge we accumulated during the laboratories. Moreover, the project introduced us to the field of audio classification, which was a new challenge for us, and we learned how to process the audio signals using the *librosa* library. A problem we faced was how to improve the network, we were not sure if we should try first to increase the network width or the depth, or what number of filters we should try because in our opinion there are a lot of decisions that we can make at the same time. Therefore, one of the main problems was the time management: to decide which experiments to perform, such that we could obtain a meaningful result.

References

[1] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *CoRR*, vol. abs/1608.04363, 2016.

[2] A. Mesaros, T. Heittola, O. Dikmen, and T. Virtanen, "Sound event detection in real life recordings using coupled matrix factorization of spectral representations and class activity annotations," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 151–155, 2015.

[3] J. Salamon and J. P. Bello, "Unsupervised feature learning for urban sound classification," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 171–175, 2015.

[4] J. Salamon and J. P. Bello, "Feature learning with deep scattering for urban sound analysis," in *2015 23rd European Signal Processing Conference (EUSIPCO)*, pp. 724–728, 2015.

[5] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, "Polyphonic sound event detection using multi label deep neural networks," in *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, 2015.

[6] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan, "Speech recognition using deep neural networks: A systematic review," *IEEE Access*, vol. 7, pp. 19143–19165, 2019.

[7] N. Ndou, R. Ajoodha, and A. Jadhav, "Music genre classification: A review of deep-learning and traditional machine-learning approaches," in *2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, pp. 1–6, 2021.

[8] A. Bansal and N. K. Garg, "Environmental sound classification: A descriptive review of the literature," *Intelligent Systems with Applications*, vol. 16, p. 200115, 2022.

[9] B. Zhu, C. Wang, F. Liu, J. Lei, Z. Lu, and Y. Peng, "Learning environmental sounds with multi-scale convolutional neural network," *CoRR*, vol. abs/1803.10219, 2018.

[10] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 1041–1044, ACM, 2015.

[11] S. Abdoli, P. Cardinal, and A. L. Koerich, "End-to-end environmental sound classification using a 1d convolutional neural network," *CoRR*, vol. abs/1904.08990, 2019.

[12] Y. Zhang, J. Zeng, Y. Li, and D. Chen, "Convolutional neural network-gated recurrent unit neural network with feature fusion for environmental sound classification," *Automation and Control in Computer Science*, vol. 55, p. 311â€"318, 2021.

[13] S. R. Kaiming He, Xiangyu Zhang and J. Sun, "Deep residual learning for image recognition," arXiv, 2015.

[14] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013.

[15] J. Sang, S. Park, and J. Lee, "Convolutional recurrent neural networks for urban sound classification using raw waveforms," in *2018 26th European Signal Processing Conference (EUSIPCO)*, pp. 2444–2448, 2018.

[16] L. Nanni, G. Maguolo, S. Brahnam, and M. Paci, "An ensemble of convolutional neural networks for audio classification," *Applied Sciences*, vol. 11, p. 5796, jun 2021.

[17] Z. Mushtaq, S.-F. Su, and Q.-V. Tran, "Spectral images based environmental sound classification using cnn with meaningful data augmentation," *Applied Acoustics*, vol. 172, p. 107581, 2021.

[18] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[19] S. S. Stevens, V. John, and E. B. Newman, "A scale for the measurement of the psychological magnitude pitch," *The Journal of the Acoustical Society of America*, vol. 8, pp. 185–190, Jan. 1937.

[20] F. Zheng, G. Zhang, and Z. Song, "Comparison of different implementations of mfcc," *Journal of Computer Science and Technology*, vol. 16, pp. 582–589, Nov. 2001.

[21] Y. Aytar, C. Vondrick, and A. Torralba, "Soundnet: Learning sound representations from unlabeled video," 2016.

[22] Y. Zhao, J. Hessel, Y. Yu, X. Lu, R. Zellers, and Y. Choi, "Connecting the dots between audio and text without parallel data through visual knowledge transfer," *CoRR*, vol. abs/2112.08995, 2021.