22nd February 2023

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

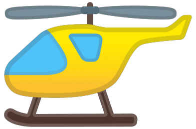
HUMAN DATA ANALYTICS PROJECT PRESENTATION

ELAHEH AHMADIESLAMLOO

GEORGE P. PRODAN

INTRODUCTION

Environmental Sound Classification

Motivation

With the rise of audio-based technologies the creation of efficient algorithms for ESC is becoming a crucial issue

Objective

Our goal is to implement CNN models to solve the ESC problem

Contributions

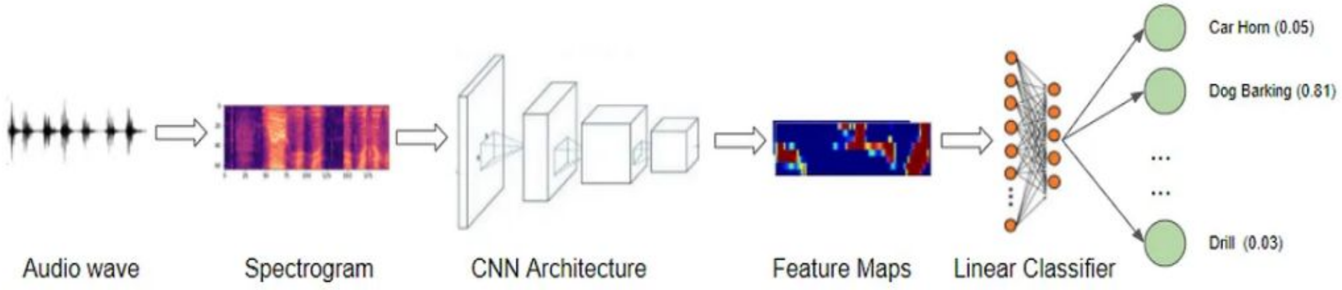
- Performance study of different architectures
- Model scalings
- Study of regularization techniques

Why CNN?

Various new data data augmentation techniques

Extract features

Successful performance in a variety of audio classification tasks



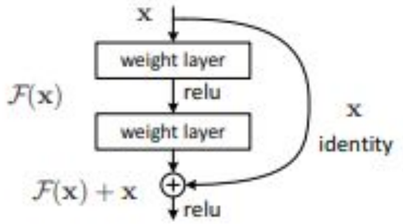
RELATED WORK

Convolutional Neural Networks

Study	Method	Dataset	Accuracy
Lee	Multiscale CNN	DCASE2017	63.4%
Piczak	Temporal CNN + attention	ESC-10	72.2%
Abdoli	1D CNN	UrbanSound8k	89%
Zhang	CNN-Gated Recurrent Unit	ESC-10	92.3%
Zhang	CNN-Gated Recurrent Unit	ESC-50	87.43%
Zhang	CNN-Gated Recurrent Unit	UrbanSound8K	96.1%

Residual Layers ⟵ ResNet Architectures

S. R. Kaiming He, Xiangyu Zhang and J. Sun, *Deep residual learning for image recognition*, 2015.



Recurrent CNNs

J. Sang, S. Park, and J. Lee, *Convolutional recurrent neural networks for urban sound classification using raw waveforms*, 2018.

Data augmentation

Improve the performance

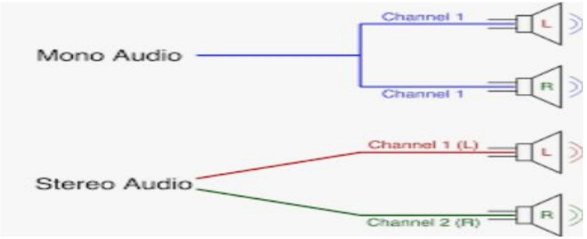
Learn more patterns (not in original)

Techniques:

- ❖ **Speed changing**
- ❖ **pitch shifting**
- ❖ **time shifting**
- ❖ **Clipping**
- ❖ **time masking,frequency masking,inserting Gaussian noise**

Main steps

1. **Data Pre-processing(mono,44100 hz)**
2. **Data Loading(subclass of tf.keras.Sequence)**
3. **Model Training (40-100 epochs)**
4. **Model Testing**



AUDIO REPRESENTATIONS

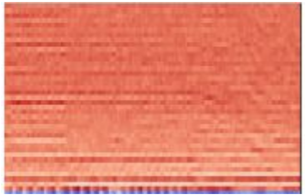
Raw audio waveforms

the raw waveforms are loaded using *librosa* library

mono audio & 44.1 kHz

Feature extraction :

- Mel spectrogram
- Mel-frequency Cepstral Coefficients (MFCCs)



DATASET: ESC-50

ESC-50

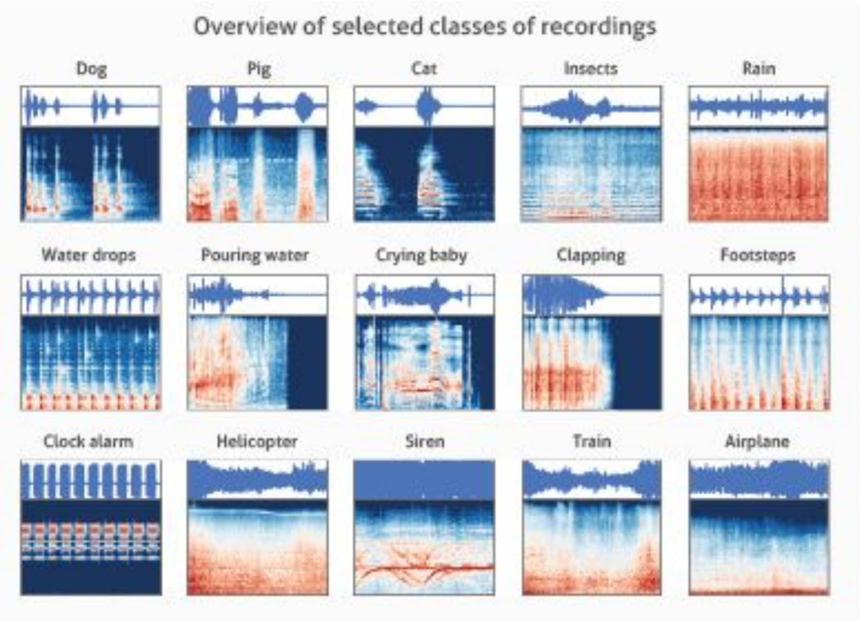
50 classes of sounds from various environments

5 second clips, 40 clips/class → 2000 clips

Split the dataset into 80/10/10

Stratified split

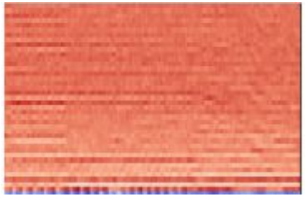
class distribution in the three sets is representative of the class distribution in the whole dataset



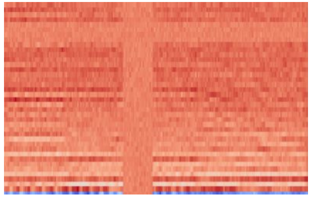
LEARNING FRAMEWORK

Load waveform

Compute MFCCs

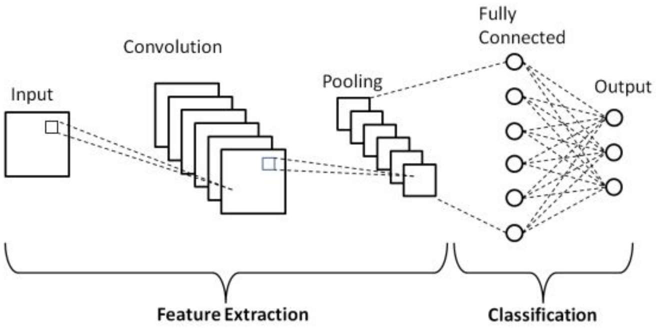


Data augmentation

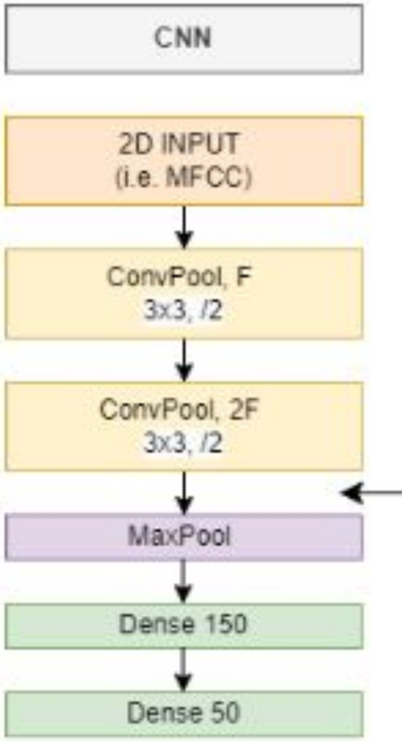


Training

Testing



Basic CNN



extra layers added here

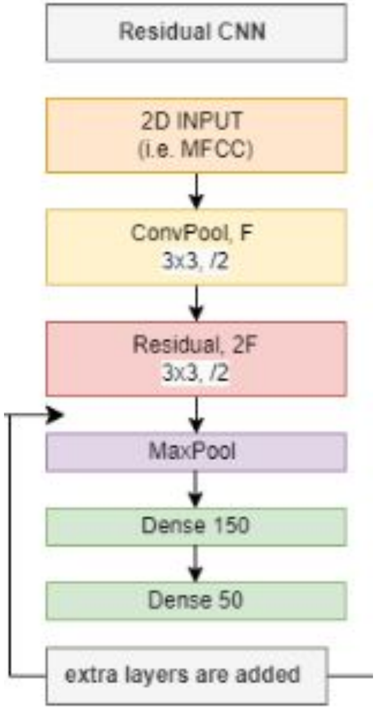
F = number of filters

CNN + Residual Layers

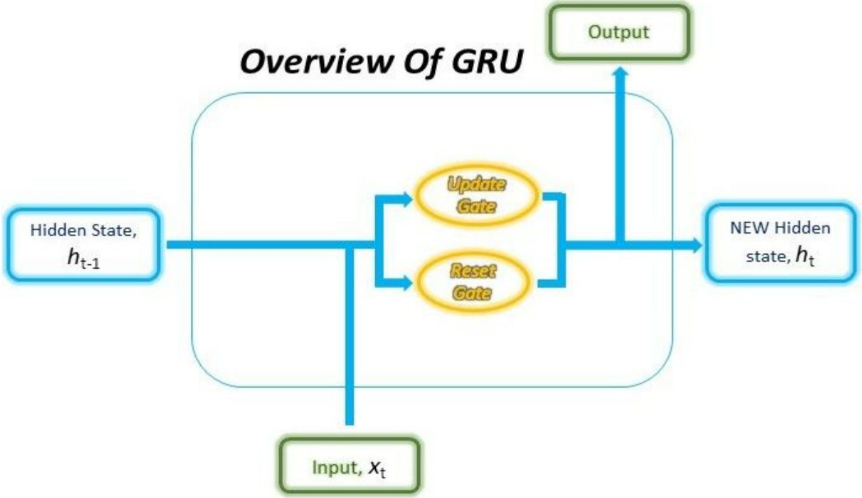
Residual Layers

significant advancement in deep neural networks for image processing tasks

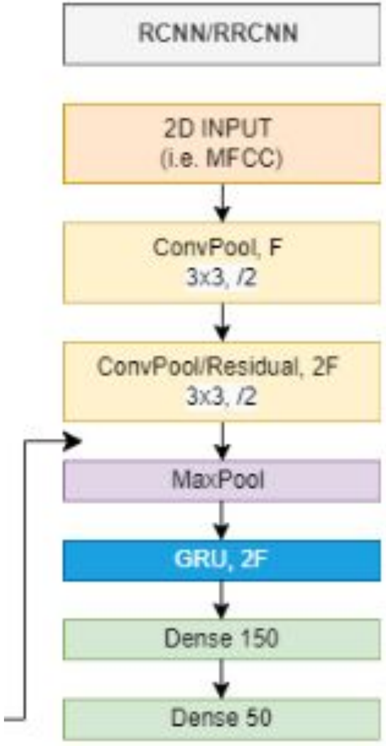
solved the problem of "gradient vanishing" in deep networks with many stacked layers



Recurrent CNNs



extra layers added here



FIRST EXPERIMENTS

performed on a CNN with 2 layers

Kernel Size

3x3	5x5	7x7
88.5%	84.5%	87.5%

Pool Size

2x2	4x4	8x8
88.5%	84%	79%

Width Scaling

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

Trying Residual Layers, or GRUs

Test Accuracy (2L)

CNN	88.5%
CNN+Residual Layers	86.5%
CNN+GRU	93.0%
CNN+Residual Layers+GRU	91.5%

DEPTH SCALING

Architecture	2L	3L	4L	5L	6L
CNN	88.5%	90.0%	91.5%	90.5%	91.0%
CNN+Residual Layers	86.5%	91.5%	87.5%	89.0%	89.0%
CNN+GRU	93.0%	92.0%	88.5%	92.0%	89.0%
CNN+Residual Layers+GRU	91.5%	92.0%	93.5%	92.0%	92.0%

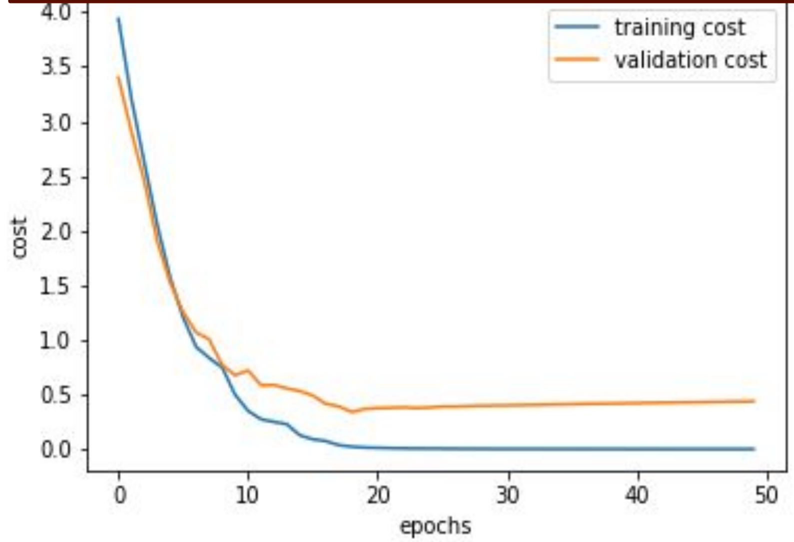
REGULARIZATION TECHNIQUES

- Dropout Layers
- Data Augmentation
  - Time Masking
  - Frequency Masking
  - Gaussian Noise

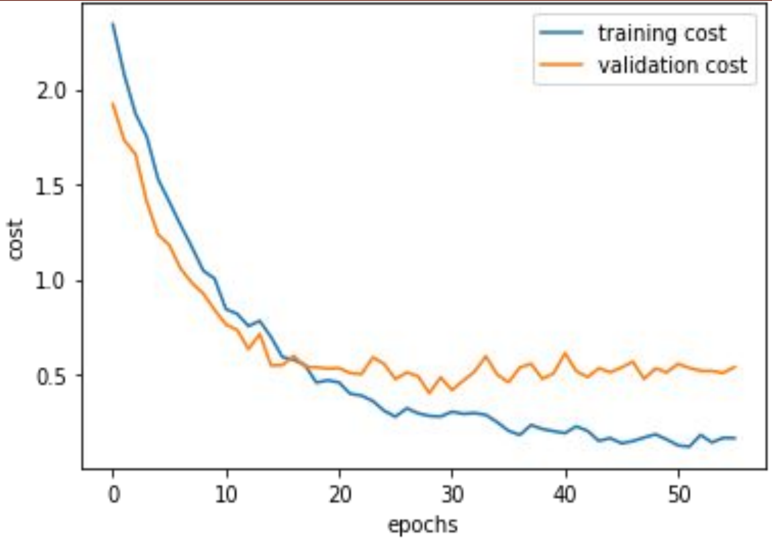
Grid search results

- Using both frequency and time masks
- 4% gaussian noise
- 0.5 dropout

without regularization



with regularization



BEST RESULTS

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

WORST RESULTS

(on a CNN with 2 layers)

Not using BN: 79.5%

Using larger LR:

2% for 0.1

15% for 0.01

Using only one FC layer: 10.5%

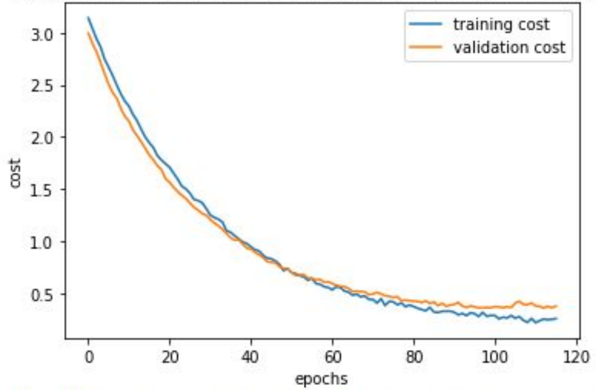
DEMO

```
class CNN(tf.keras.Model):
    def __init__(self, F=32, n=2, num_classes=50, p_dropout=None, residual=False):
        super().__init__()
        self.F = F
        self.n = n
        self.p_dropout = p_dropout
        self.residual = residual

        Layer = ResBlock2D if residual else ConvPool2D

        activation = 'relu'
        self.description = f'CNN_{n}layers'
        hidden_units = 150

        if residual:
          self.identity_1 = Identity(F)
          if n > 2:
            self.identity_2 = Identity(F)


        # 1st block
        self.layer_1 = ConvPool2D(F)

        self.layer_2 = Layer(F*2, maxpool=True)
        if n > 2:
          if residual: hidden_units = 500
          self.layer_3 = Layer(F*2, maxpool=True)
        if n > 3:
          hidden_units = 450
          self.layer_4 = Layer(F*2, maxpool=False)

        if n > 4:
          self.layer_5 = Layer(F*2, maxpool=False, padding="same")
        if n > 5:
          self.layer_6 = Layer(F*2, maxpool=False, padding="same")
        k = 10 if n == 2 else 5
        self.pool_gru = tf.keras.layers.MaxPool2D((k, 1), strides=(k, 1), padding='same')
```

DEMO

```
[5.830000 min, epoch 115] Train Cost: 0.222256 --- Validation Cost: 0.382603 --- Train accuracy: 0.963750 --- Validation accuracy: 0.915000
[5.880000 min, epoch 116] Train Cost: 0.240868 --- Validation Cost: 0.378268 --- Train accuracy: 0.960625 --- Validation accuracy: 0.910000
[5.920000 min, epoch 117] Train Cost: 0.256871 --- Validation Cost: 0.363395 --- Train accuracy: 0.946875 --- Validation accuracy: 0.915000
[5.980000 min, epoch 118] Train Cost: 0.250522 --- Validation Cost: 0.378364 --- Train accuracy: 0.951250 --- Validation accuracy: 0.920000
[6.040000 min, epoch 119] Train Cost: 0.253783 --- Validation Cost: 0.364822 --- Train accuracy: 0.948125 --- Validation accuracy: 0.920000
[6.090000 min, epoch 120] Train Cost: 0.260722 --- Validation Cost: 0.380681 --- Train accuracy: 0.953125 --- Validation accuracy: 0.910000
```

CNN_2layers_50_E120_OAdam_LR0.0001_pD0.5_A0.4_feb13_t2240




```
[TESTING] Test accuracy: 0.930000 --- Test cost: 0.268969
```

DEMO

TIME	MODEL NAME	INPUT	BATCH	EPOCHS	AUGMENT PRO	DROPOUT PRO	OPTIMIZER	LR	TRAIN ACC	VAL ACC	TEST ACC	BEST EPOCH	PARAMETERS	FILTERS	RESIDUAL	TIME/EPOCH
02-13 13:45	CNN_6layersGR	mfcc	50	100	0		Adam	0.0001	1	0.93	0.92	87	6267086	128	TRUE	4.3742
02-13 13:57	CNN_6layersGR	mfcc	50	100	0		Adam	0.0001	1	0.915	0.89	69	3210702	128	FALSE	3.0054
02-13 14:12	CNN_5layersGR	mfcc	50	100	0		Adam	0.0001	1	0.925	0.92	61	2619598	128	FALSE	2.8041
02-13 14:28	CNN_4layersGR	mfcc	50	100	0		Adam	0.0001	1	0.905	0.885	60	2028494	128	FALSE	2.8686
02-13 14:45	CNN_3layersGR	mfcc	50	100	0		Adam	0.0001	1	0.92	0.92	79	1329940	128	FALSE	2.882
02-13 15:15	CNN_3layersGR	mfcc	50	100	0		Adam	0.0001	1	0.92	0.915	82	1329940	128	FALSE	2.754
02-13 15:21	CNN_2layersGR	mfcc	50	100	0		Adam	0.0001	1	0.93	0.93	100	738836	128	FALSE	2.4178
02-13 15:33	CNN_2layers	mfcc	50	100	0		Adam	0.0001	0.9988	0.915	0.865	53	3004436	128	TRUE	2.2961
02-13 15:44	CNN_2layers	mfcc	50	60	0		Adam	0.0001	0.9994	0.92	0.885	60	2302484	128	FALSE	1.347833333
02-13 16:20	CNN_3layers	mfcc	50	60	0		Adam	0.0001	1	0.92	0.915	30	5653710	128	TRUE	1.723166667
02-13 16:25	CNN_4layers	mfcc	50	60	0		Adam	0.0001	0.9956	0.93	0.875	28	6835918	128	TRUE	1.878833333
02-13 16:30	CNN_5layers	mfcc	50	60	0		Adam	0.0001	0.9994	0.94	0.89	25	8018126	128	TRUE	2.098666667
02-13 16:43	CNN_6layers	mfcc	50	60	0		Adam	0.0001	0.9956	0.915	0.885	26	9200334	128	TRUE	2.355333333

CONCLUSIONS

- **Features such as BNs and GRUs improve the performance**
- **It is not always helpful to have more layers**
- **Data Augmentation can improve the results by 1-2%**
- **Residual Layers without GRU are not a better choice**

Thank you for your attention!