

Audio Classification - UrbanSound8k

Neste relatório, descreveremos a abordagem adotada para classificar dados sonoros urbanos usando técnicas de Deep Learning.

O problema de classificação consiste em categorizar dados sonoros urbanos em uma das dez classes pré-definidas, sendo essas dez classes: ar condicionado, buzina de um carro, crianças a brincar, ladrar de um cão, perfuração, motor a trabalhar, tiro de uma arma, martelo pneumático, sirene e música de rua.

A primeira abordagem passou por treinar uma CNN de duas dimensões. Acharmos que esta seria a melhor solução já que as mfccs obtidos com a ajuda da biblioteca Librosa têm uma shape que se assemelha ao de uma imagem e para problemas com imagens é comum usar-se CNNs com duas dimensões. A segunda abordagem foi uma RNN, mais

precisamente um LSTM unidirecional. Optamos por esta solução porque, apesar de usarmos como input os mfccs e não o som em si, as LSTMs são muito usadas para fazer speech recognition, que é um problema que pensamos que poderia ser semelhante ao que nos foi proposto.

No que toca ao pre-processing começamos por verificar que apenas duas (gun_shot e car_horn) das dez classes não têm cerca de 11% de representatividade no dataset, tendo 4,3% e 4,9% respetivamente, como podemos ver na figura 1. Devido à diferença destes valores não ser muito elevada decidimos não usar nenhuma técnica para tratar de datasets desbalanceados, como resampling ou loss functions com pesos diferentes para cada classe. De seguida notamos que, apesar de grande parte dos sons terem quatro segundos de duração, havia ainda um número razoável dos sons que não tinham essa duração, como podemos ver na figura 2. Para resolver este problema resolvemos repetir o som até que ficasse com os 4 segundos. Isto pode ser bom pois dá mais oportunidades ao modelo de tirar ilações sobre aquele som visto que ele se repete várias vezes nos quatro segundos, mas também pode levar o modelo a tirar conclusões erradas, como por exemplo, como há muitos sons da classe gun_shot com menos que 4 segundos, o modelo pode considerar que deve classificar um som como gun_shot se a certa altura ele se começar a repetir.

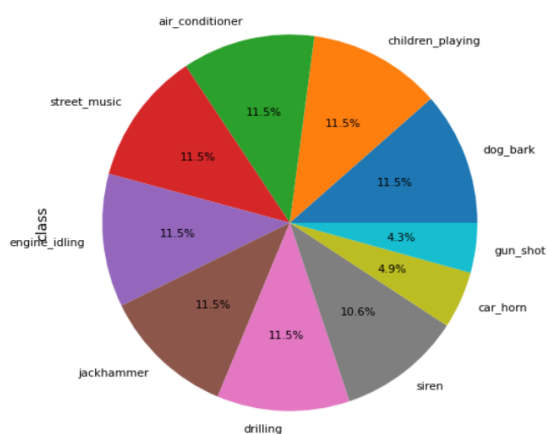


Figura 1

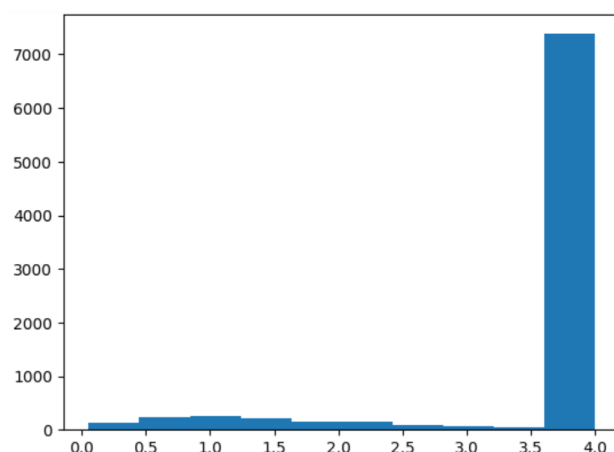


Figura 2

Analizamos também uma onda de um exemplo de cada classe para termos uma ideia de que classes teriam ondas parecidas e poderiam ser confundidas pelo modelo.

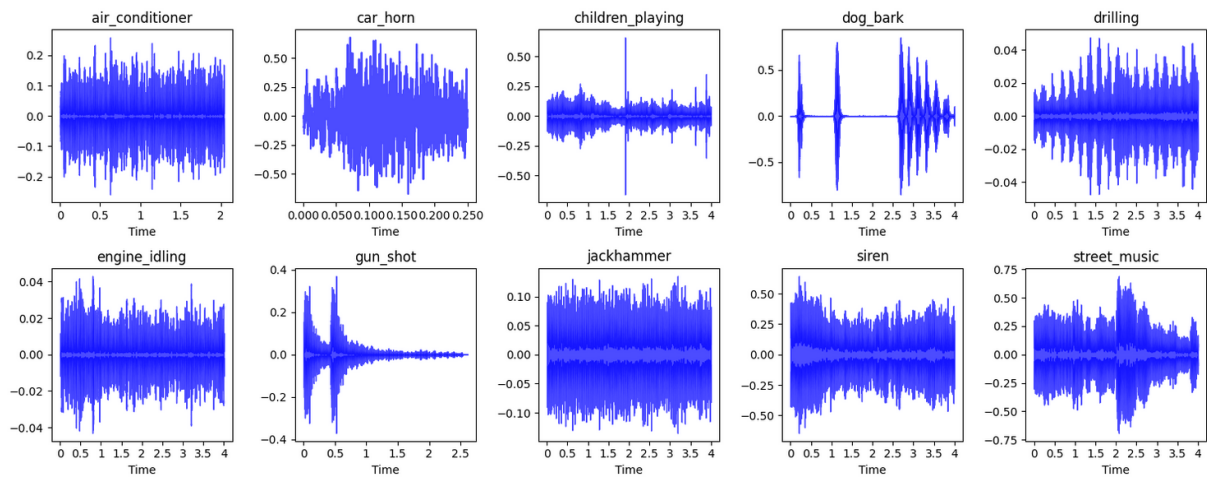


Figura 3

Antes de passarmos à obtenção dos mfccs adicionamos um ruído com um desvio padrão de 0.008 a todos os sons. Após vários testes concluímos que este valor para o desvio padrão alterava o som de maneira suficiente sem que o tornasse imperceptível, como se vê nas figuras 4 e 5. As figuras mostram o som antes e depois de ser acrescentado o ruído.

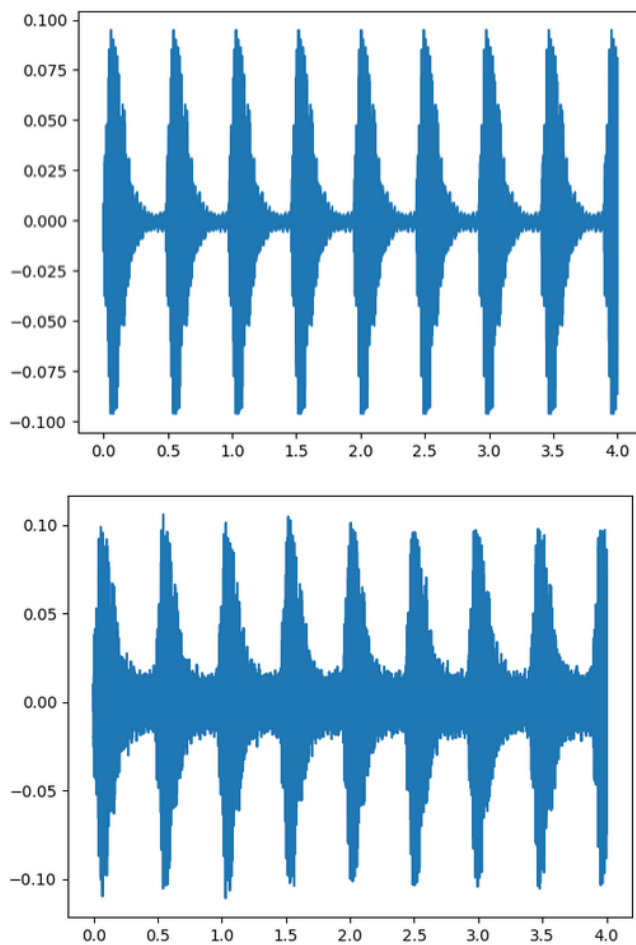


Figura 4

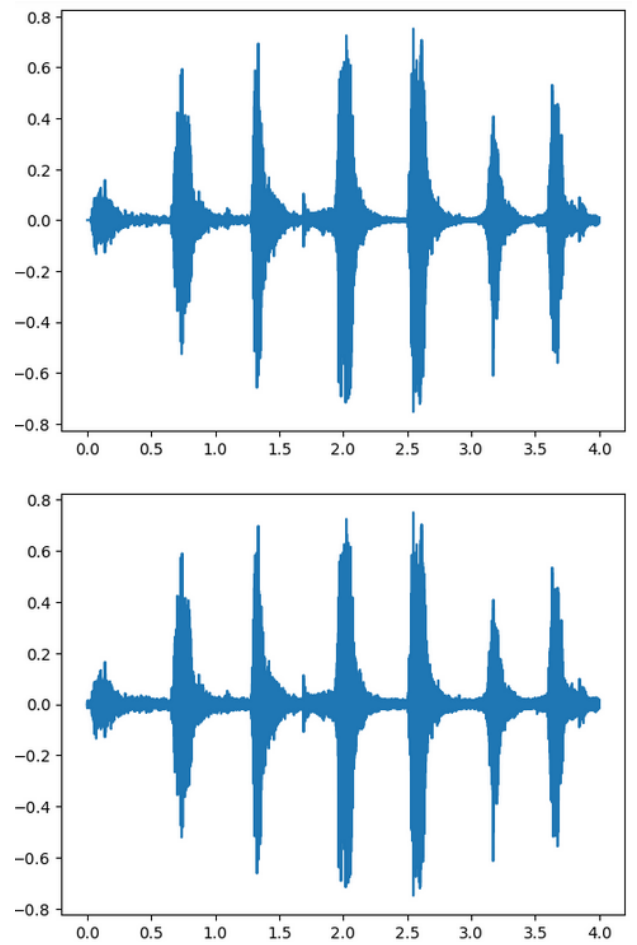


Figura 5

No que toca à extração dos mfccs começamos por alterar a frequência de todos os sons para 44100 Hz. Depois obtivemos os mfccs com a ajuda da biblioteca Librosa e guardamos um array com os mfccs e outro com as classes nas respectivas pastas.

Já no treino dos classificadores, começamos por definir que iríamos usar a `sparse_categorical_crossentropy` como a função de loss e a `sparse_categorical_accuracy` como métrica de avaliação. Para avaliar a performance dos classificadores usamos o sistema de cross validation sugerido no website do dataset UrbanSound.

Quanto ao otimizador utilizamos o adam para também para ambos os modelos, apesar de termos explorado a possibilidade de o stochastic gradient descent para a LSTM tentando descobrir a learning rate ótima para treinar o modelo. Como pode ser visto na figura 6, a alteração da learning rate (eixo x) não alterou a loss (eixo y) o que nos fez usar o otimizador adam.

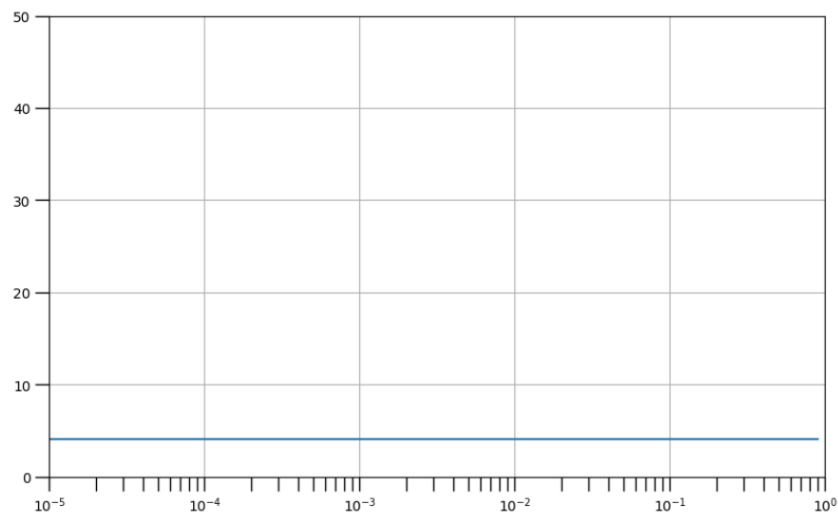


Figura 6

Optamos pelo uso da função `EarlyStopping` para monitorizar a accuracy do teste para tornar o treino mais rápido. Esta função para o treino se houver 10 epochs em que a accuracy do teste não aumentou pelo menos 0.001.

Foram treinados modelos com os sons que estavam no foreground, no background e um modelo com todos os sons. O número de epochs foi 50, embora os treinos tenham acabado antes devido ao early stopping, e o tamanho das mini batches foi 64.

Não foi usado nenhum tipo de data augmentation por considerarmos não ser produtivo usar os mesmos métodos que são usados nas imagens, como rodar a imagem ou fazer zoo visto que uma onda sonora tem sempre o mesmo aspeto e está sempre na mesma posição.

Ambos os modelos têm layers de Dropout para tentar minimizar o overfitting e de Batch Normalization para agilizar o tempo de treino e melhorar a sua estabilidade.

No caso da CNN usamos como input os vetores com os mfccs com um shape de (40, 321) que tínhamos obtido previamente.

Obtivemos os seguintes resultados para cada treino.

	2D-CNN	2D-CNN_FG	2D-CNN_BG
0	63.573885	73.794210	50.597608
1	64.189190	77.956986	43.333334
2	54.378378	60.508472	51.044774
3	63.737375	67.741936	54.861110
4	74.252135	77.677226	61.172163
5	62.940460	74.174756	54.220778
6	69.928402	81.558937	58.333331
7	67.617863	75.543481	54.724407
8	68.014705	83.818179	41.353384
9	71.087217	85.387325	48.698884

Figura 7

Como pode ser visto na figura 7, onde cada linha corresponde a uma fold de teste, os melhores resultados foram no modelo que foi treinado com os sons no foreground e os piores no modelo treinado com os sons no background. Isto é expectável pois, tal como o ser humano, o modelo dá mais ênfase aos sons que estão perto de si, neste caso mais perto do microfone que gravou o áudio.

Pegando no melhor resultado para cada dataset, podemos verificar com a ajuda dos gráficos das figuras 8, 9 e 10 que, apesar dos nossos esforços, acabou por haver um pouco de overfitting.

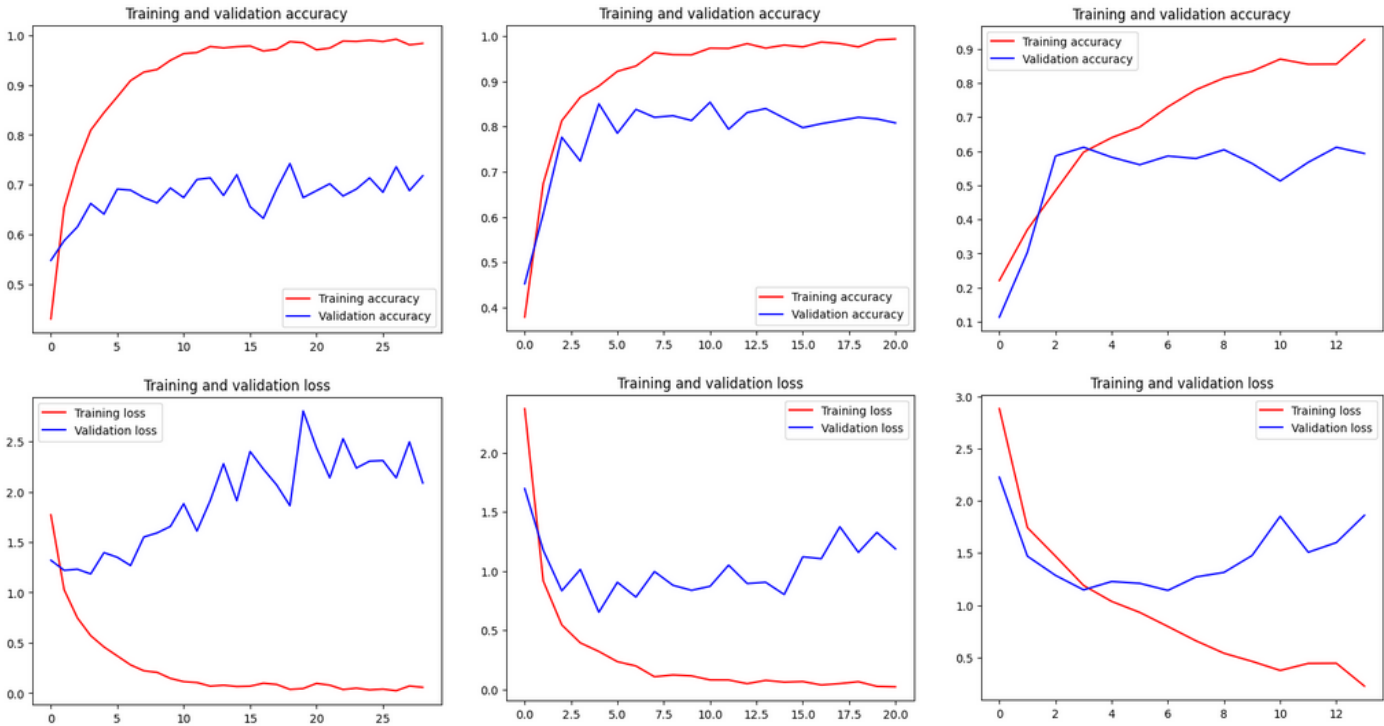


Figura 8 - All Sounds

Figura 9- Foreground

Figura 10 - Background

Analisando as matrizes confusão dos resultados dos treinos podemos verificar que treinar o modelo com todos os sons teve uma accuracy de 65.97% e um desvio padrão de 5.24%, treinar com os sons no foreground teve uma accuracy de 75.82% e um desvio padrão de 7.09% e treinar com os sons no background teve uma accuracy de 51.82% e um desvio padrão de 5.89%.

All Sounds

air_conditioner	443	11	46	50	43	109	2	134	22	140
car_horn	8	327	11	8	11	12	0	10	3	39
children_playing	31	3	614	69	17	50	5	12	62	137
dog_bark	14	10	85	778	14	22	3	2	30	42
drilling	32	21	28	53	678	17	2	90	40	39
engine_idling	80	9	57	37	89	568	3	69	43	45
gun_shot	0	0	9	7	2	4	342	9	0	1
jackhammer	79	1	24	5	143	96	0	529	79	44
siren	7	2	68	21	3	23	0	7	752	46
street_music	36	20	96	17	19	34	0	13	43	722

Figura 11

Foreground

air_conditioner	333	0	29	0	43	81	14	36	0	33
car_horn	0	135	6	1	2	0	0	0	3	6
children_playing	9	4	472	24	24	17	7	1	4	26
dog_bark	3	1	38	567	8	6	3	1	7	11
drilling	17	6	23	28	726	33	5	38	4	22
engine_idling	67	0	39	5	52	664	2	87	0	0
gun_shot	7	0	4	2	3	2	284	0	0	2
jackhammer	57	0	4	0	100	49	2	490	0	29
siren	3	0	31	10	1	3	1	0	194	26
street_music	23	4	61	5	35	11	5	6	22	453

Figura 12

Background

air_conditioner	202	13	69	40	9	6	7	33	13	39
car_horn	10	188	20	5	4	4	1	8	22	14
children_playing	39	26	171	68	3	2	2	15	39	47
dog_bark	23	14	32	170	3	3	1	12	77	20
drilling	8	13	24	3	13	1	2	18	15	1
engine_idling	12	3	6	9	1	7	0	9	29	15
gun_shot	6	0	1	3	42	0	56	0	2	0
jackhammer	32	8	35	14	17	3	1	102	73	11
siren	6	10	48	38	4	2	2	10	523	4
street_music	34	25	79	17	0	3	0	2	12	142

Figura 13

Comparando a matriz confusão do treino com todos os sons com a soma das matrizes de confusão dos outros dois treinos podemos ver que o primeiro caso acertou 5753 vezes enquanto que o segundo acertou 5892 vezes.

All Sounds - 5753

air_conditioner	443	11	46	50	43	109	2	134	22	140
car_horn	8	327	11	8	11	12	0	10	3	39
children_playing	31	3	614	69	17	50	5	12	62	137
dog_bark	14	10	85	778	14	22	3	2	30	42
drilling	32	21	28	53	678	17	2	90	40	39
engine_idling	80	9	57	37	89	568	3	69	43	45
gun_shot	0	0	9	7	2	4	342	9	0	1
jackhammer	79	1	24	5	143	96	0	529	79	44
siren	7	2	68	21	3	23	0	7	752	46
street_music	36	20	96	17	19	34	0	13	43	722

Figura 14

Foreground + Background - 5892

air_conditioner	535	13	98	40	52	87	21	69	13	72
car_horn	10	323	26	6	6	4	1	8	25	20
children_playing	48	30	643	92	27	19	9	16	43	73
dog_bark	26	15	70	737	11	9	4	13	84	31
drilling	25	19	47	31	739	34	7	56	19	23
engine_idling	79	3	45	14	53	671	2	96	29	15
gun_shot	13	0	5	5	45	2	340	0	2	2
jackhammer	89	8	39	14	117	52	3	592	73	40
siren	9	10	79	48	5	5	3	10	717	30
street_music	57	29	140	22	35	14	5	8	34	595

Figura 15

Já na LSTM usamos como input um vetor com um shape de (40) em que cada ponto é a média dos valores ao longo do segundo eixo do vetor dos mfccs. Apesar de poder parecer pouco intuitivo, após alguns testes este foi o formato que nos deu os resultados menos maus (Figura 13).

	LSTM	LSTM_FG	LSTM_BG
0	48.797250	56.591642	35.458168
1	41.328830	46.594983	33.030304
2	41.621622	42.881355	43.582091
3	40.808082	40.143371	41.203704
4	51.923078	54.147816	45.054945
5	39.854193	53.398061	31.493506
6	53.937948	54.182512	41.987181
7	45.657569	50.905800	34.251967
8	54.534316	67.090911	32.330826
9	42.771804	54.929578	29.739776

Figura 16

Analisando os gráficos equivalentes aos do treino com a CNN, vemos que houve mais overfitting do que o treino anterior.

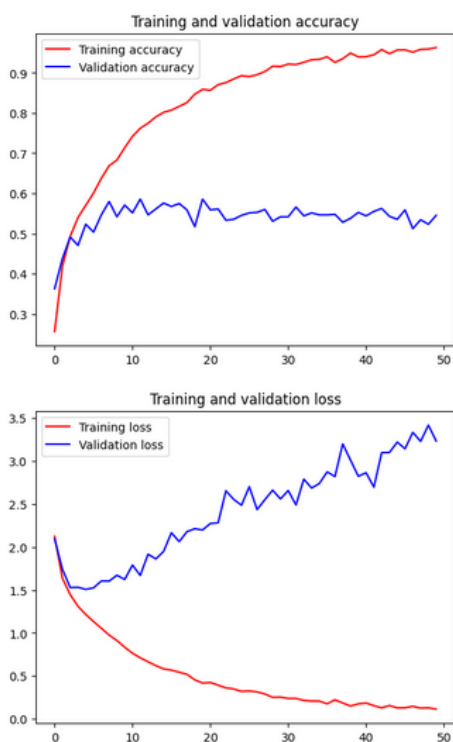


Figura 17 - All Sounds

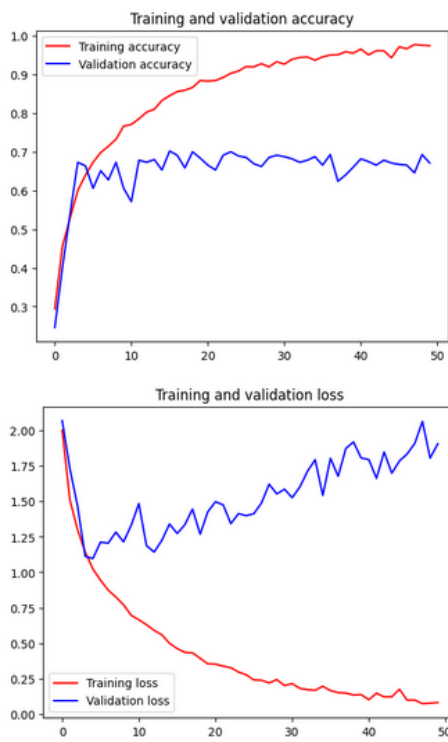


Figura 18 - Foreground

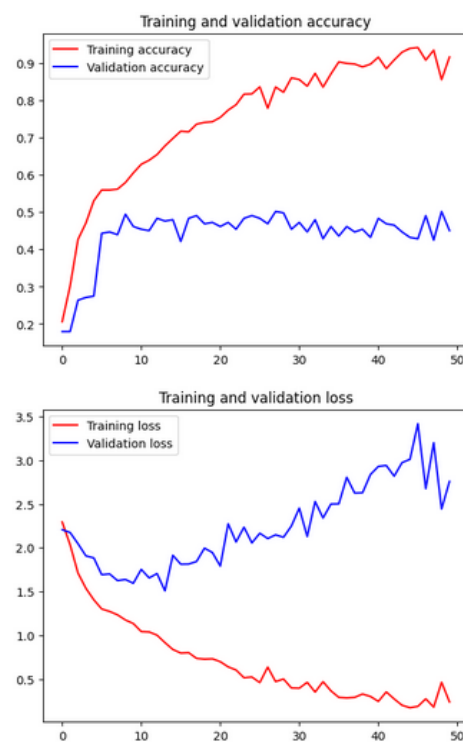


Figura 19 - Background

Analisando as matrizes confusão dos resultados dos treinos podemos verificar que treinar o modelo com todos os sons teve uma accuracy de 46.12% e um desvio padrão de 5.43%, treinar com os sons no foreground teve uma accuracy de 52.09% e um desvio padrão de 7.23% e treinar com os sons no background teve uma accuracy de 36.81% e um desvio padrão de 5.3%.

All Sounds

air_conditioner	163	38	88	47	88	105	17	108	37	309
car_horn	7	287	9	14	40	11	0	11	5	45
children_playing	31	8	398	211	32	48	67	18	62	125
dog_bark	15	11	146	605	32	29	25	3	62	72
drilling	18	33	56	91	461	37	7	152	39	106
engine_idling	61	22	110	75	57	365	1	78	106	125
gun_shot	1	0	47	42	10	1	265	1	2	5
jackhammer	35	31	63	9	334	65	0	308	18	137
siren	23	15	52	176	24	14	2	11	575	37
street_music	52	40	102	53	71	26	7	31	26	592
Predicted Label	air_conditioner	car_horn	children_playing	dog_bark	drilling	engine_idling	gun_shot	jackhammer	siren	street_music

Figura 20

Foreground

air_conditioner	127	0	53	19	41	60	0	57	15	197
car_horn	1	131	0	1	12	1	0	1	0	6
children_playing	14	1	307	112	33	32	35	8	8	38
dog_bark	9	1	83	466	14	10	13	3	18	28
drilling	18	9	41	49	486	39	13	127	13	107
engine_idling	94	2	162	29	61	393	0	94	23	58
gun_shot	0	0	33	31	18	2	213	3	0	4
jackhammer	28	0	9	5	212	72	0	307	1	97
siren	11	2	16	32	5	6	0	0	156	41
street_music	37	8	34	39	60	11	5	27	20	384
Predicted Label	air_conditioner	car_horn	children_playing	dog_bark	drilling	engine_idling	gun_shot	jackhammer	siren	street_music

Figura 21

Background

air_conditioner	44	34	70	97	9	3	13	10	58	93
car_horn	10	165	4	28	6	9	1	3	14	33
children_playing	34	5	135	91	3	5	1	6	44	88
dog_bark	11	23	55	147	4	3	5	13	60	34
drilling	10	6	3	13	12	0	0	15	25	14
engine_idling	8	10	1	46	0	0	0	1	4	14
gun_shot	0	2	8	8	39	2	61	8	1	0
jackhammer	10	10	15	51	15	21	8	112	69	63
siren	29	9	35	139	7	3	7	3	339	14
street_music	16	31	67	29	7	1	6	13	8	108
Predicted Label	air_conditioner	car_horn	children_playing	dog_bark	drilling	engine_idling	gun_shot	jackhammer	siren	street_music

Figura 22

Comparando a matriz confusão do treino com todos os sons com a soma das matrizes de confusão dos outros dois treinos podemos ver que o primeiro caso acertou 4019 vezes enquanto que o segundo acertou 4096 vezes.

All Sounds - 4019

air_conditioner	163	38	88	47	88	105	17	108	37	309
car_horn	7	287	9	14	40	11	0	11	5	45
children_playing	31	8	398	211	32	48	67	18	62	125
dog_bark	15	11	146	605	32	29	25	3	62	72
drilling	18	33	56	91	461	37	7	152	39	106
engine_idling	61	22	110	75	57	365	1	78	106	125
gun_shot	1	0	47	42	10	1	265	1	2	5
jackhammer	35	31	63	9	334	65	0	308	18	137
siren	23	15	52	176	24	14	2	11	575	37
street_music	52	40	102	53	71	26	7	31	26	592
Predicted Label	air_conditioner	car_horn	children_playing	dog_bark	drilling	engine_idling	gun_shot	jackhammer	siren	street_music

Figura 23

Foreground + Background - 4096

air_conditioner	171	34	123	116	50	63	13	67	73	290
car_horn	11	299	4	29	18	10	1	4	14	39
children_playing	48	6	442	203	36	37	36	14	52	126
dog_bark	20	24	138	613	18	13	18	16	78	62
drilling	28	15	44	62	498	39	13	142	38	121
engine_idling	102	12	163	75	61	393	0	95	27	72
gun_shot	0	2	41	39	57	4	274	11	1	4
jackhammer	38	10	24	56	227	93	8	419	70	160
siren	40	11	51	171	12	9	7	3	495	55
street_music	53	39	101	68	67	12	11	40	28	492
Predicted Label	air_conditioner	car_horn	children_playing	dog_bark	drilling	engine_idling	gun_shot	jackhammer	siren	street_music

Figura 24

Tendo em conta os nossos testes, concluímos que uma CNN resolve com alguma qualidade este problema enquanto uma LSTM pode ter mais dificuldades. Isto pode ter acontecido também devido a termos usado uma média dos valores dos mfccs e termos perdido informação no processo. Outra explicação pode ser a rede LSTM que usamos não ser a mais adequada, precisando por exemplo de mais camadas e mais unidades dentro de cada camada. Infelizmente, não temos ao nosso dispor poder computacional suficiente para executarmos testes com redes neurais de maior calibre que exijam mais tempo de execução.

Concluímos também que, havendo a possibilidade, é mais proveitoso treinar um modelo para classificar os sons que estão no foreground e outro para os sons que estão no background pois obtivemos mais acertos neste caso do que no modelo treinado com todos os sons. Outra vantagem que teria, que não foi explorada, seria usar técnicas de pre-processing diferentes para os dois tipos de sons e até mesmo modelos diferentes já que pode haver soluções melhores para um caso que para outro.

Todas as figuras podem ser vistas nos notebooks com mais detalhe.