Gun caliber classification based on gunshot audio

1st Davide Varricchio
Computer Science master student
University of Milan
Milan, Italy
davide.varricchio@studenti.unimi.it

***Abstract*—Acoustic gun caliber classification is a challenging task that could be useful in multiple fields such as law enforcement, forensic investigations and public safety. In this work K-means is used to visualize the feature space, and then multiple classification algorithms are trained on Mel-frequency Cepstrum Coefficients (MFCCs), log-mel-spectrogram and YAMNet embeddings. Simple models like K-Nearest Neighbour and Random Forest achieved 54-56% accuracy, while the more complex 2D convolutional neural networks were capable of reaching 68-75% even with a small number of layers. The best result was obtained with a 1D convolutional neural network trained on MFCCs, which achieved 80% accuracy. In the end, the MFCCs feature perfomed better, while the attempt at transfer learning with YAMNet embeddings didn't show the desired results.**

I. Introduction

Automatic gun identification by its gunshot is a useful task with applications in security, military, and forensic investigation. For example, it could help law enforcement to act quickly and accordingly, or to extract valuable information from a crime scene. In the literature, the classification task is also addressed with computer vision techniques to analyze camera frames, but this approach comes with limitations:

- Cameras have blind spots, and offenders tend to avoid the surveillance device.
- The recognition accuracy depends on light and weather condition (e.g. night or fog).

Audio data does not have these limitations, hence the interest in this method. Nonetheless, both approaches can be used jointly if possible, to improve the result. The problem has been addressed at different levels of detail in the literature, ranging from distinguishing rifles and pistols [1], all the way to the exact model of the gun [2], with increasing difficulty. In any case, complex models are often employed to face off this challenging task. For example, in [1] convolution based and transformers architecture were used. In this work, recognition of gun caliber size is performed based on popular audio features in the field, namely the log mel spectrogram and Mel-frequency Cepstrum Coefficients (MFCCs). In addition, embeddings are generated with YAMNet, a deep Convolutional Neural Network (CNN) created by Google to predict audio events from 521 classes, in an attempt at transfer learning. First, the obtained features are visualized using K-Means clustering, then they are used in K-Nearest Neighbour (KNN), Random Forest, MultiLayer Perceptron (MLP), and CNN for classification. In this way, the simpler models can provide a baseline for the more complex ones, also allowing to better compare the performance of the different features.

II. Methods and models

The data is downloaded from the Gunshot Audio Forensics Dataset [3], which includes audio of 18 firearms, recorded from 20 different positions and with 3 distinct devices, for a total of 6512 samples. After grouping by caliber of the weapon, 10 classes are found, with some of them having approximately $1/3$ of the samples of the others. The recordings were acquired with the following devices, sampled at 44,100 Hz, single channel:

- Zoom H4N.
- iPhone 7.
- Samsung Galaxy S7.

The setting in which the recording took place is described in figure 1.

A. Feature extraction and pre-processing

First, the waveform audio files are resampled at 16,000 Hz because this is the rate that the YAMNet model was trained on. This step also heavily reduces computation, and should not impact the recognition task, since gunshot's frequency is less tan 2,500 Hz [4] and the highest measurable one is 8,000 Hz (the Nyquist frequency i.e. the sampling rate halved). Then the following features are extracted:

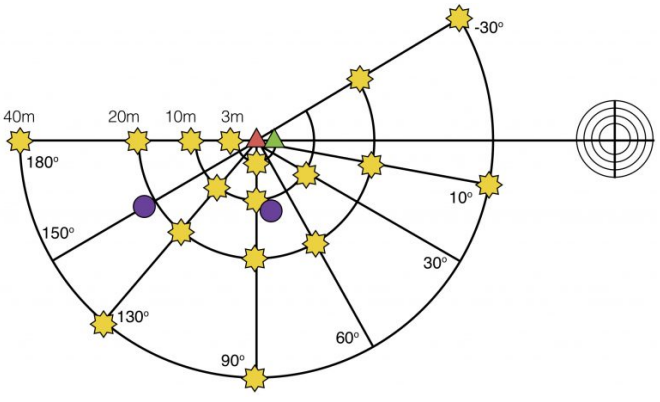

Fig. 1. Schema of the experiment that collected the data. The yellow star, purple circle and green triangle are the recording positions. The shooter was positioned at the red triangle and shot towards the indicated target

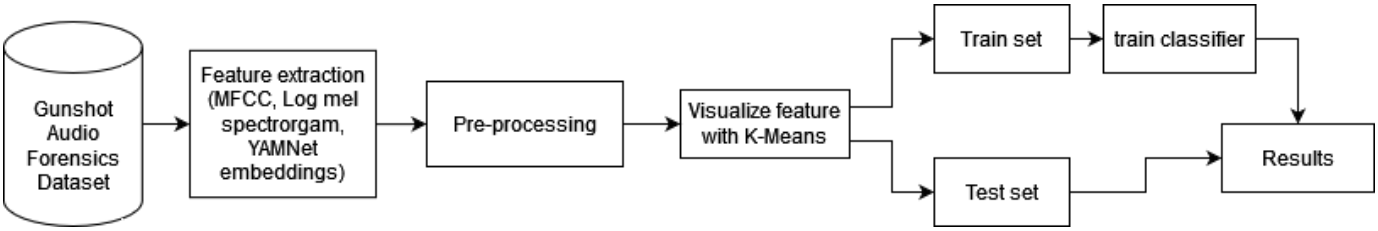

Fig. 2. Block diagram of the methodology

- **Log Mel Spectrogram**: It applies the Mel scale to the spectrogram, to better represent the non linearity of the human hearing. It is computed with the subsequent steps:
  - Windowing: the audio signal is divided in short overlapping windows (or frames), using zero padding if necessary, to obtain equal lengths. The padding is applied both on the left and right of the frame, centering the signal. In this project, the Hann function was used as a window function, with a window length of 2048 and a hop length of 512. The lengths integers refer to the number of audio signal samples.
  - Discrete Fourier Transform (DFT): the DFT is computed on every window, generating for each a vector of $1 + w/2$ coefficients, where w is the window length. The value of the kth coefficient within a vector represents the amplitude of the kf_s/w frequency bin, where f_s is the sampling frequency and $k = 0, 1, ..., w/2$. The vectors can be used in order as the columns of a matrix, and if the magnitude of the coefficients is computed, it can be treated as an image and thus visualized. This is the so called spectrogram, which shows the evolution of the signal in the time-frequency domain.
  - Mel filter banks: the powers of the spectrum (the magnitude squared) obtained above, are mapped onto the mel scale using triangular overlapping filters. In this way, the Mel Spectrogram is obtained. In this work, 128 filters are used, yielding a matrix of $128 \times n_w$ for every sample, where n_w is the number of DFT windows.
  - Log operator: apply the base 10 logarithm to the Mel Spectrogram.
- **Mel-Frequency Cepstrum Coefficients (MFCCs)**: it is popular in the field of speech processing, but has shown to be useful in other audio analysis tasks too. The coefficients are computed taking the Discrete Cosine Transform (DCT) of the Log Mel Spectrogram. Usually, only 10-20 DCT coefficients are retained. In this project, 20 MFCCs per window are used, yielding a matrix of $20 \times n_w$ for every sample, where n_w is the number of windows.
- **YAMNet embeddings**: YAMNet is a deep CNN developed by Google, that predicts 521 audio event classes from the AudioSet-YouTube corpus it was trained on. The accepted input is a single channel wave form file, sampled at 16,000 Hz, but internally, the Log Mel Spectrogram is computed, with sliding windows of length 0.96 seconds and hop 0.48 seconds. The model can also be used to extract a vector of length 1024 for every window, called embedding, which is the output of the last average-pool layer. The embeddings can then be used as features in other machine learning models. This is done on the assumption that YAMNet, being a complex model trained on millions of audio samples, is able to produce meaningful features also for other acoustic tasks.

Since the duration of the audio files varies between 1.8-2.3 seconds, all the samples are zero padded so they have the same number of windows, feature wise. Finally, the MFCCs and Log Mel Spectrogram are min-max normalized across the samples:

$$x_{norm} = \frac{x - min(x)}{max(x) - min(x)} \tag{1}$$

where x is the generic matrix element, and the min and max functions are computed over that element in all the samples. After these pre-processing steps, the holdout method is used to generate a train and test set for every feature, randomly picking 80% of the samples for training, and 20% for testing. Since the classes have unbalanced samples numbers, the splitting is done keeping the proportions of each class. Also, since the class labels are categorical, they are encoded with unique integers.

B. K-Means

K-Means (or Lloyd's algorithm) is an algorithm for the unsupervised task of clustering, i.e. grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar. The algorithm takes in input the K number of desired clusters and works as follows:

1) Select K points as the initial centroids (center of the clusters).
2) repeat:
   a) Form K clusters by assigning all points to the closest centroid.
   b) Recompute the centroid of each cluster, using the mean of the points belonging to it
3) Until the centroids don't change

The clusters obtained in this way are the result of a local optimum, since computing them exactly is an NP-hard task.

The clustering of the 3 extracted features are here visualized through Principal Component Analysis (PCA), a dimensionality reduction technique. 2 components are used, so that a plot in 2 dimension can be generated. The input number of clusters K is set to 10, like the classes obtained from the dataset. Overall, the clustering obtained with MFCCs seems to be better separated compared to the Log Mel Spectrogram ones, even if a decent amount of overlapping is present. The YAMNet embeddings on the other hand, are of difficult interpretation. This could be caused by the linear nature of the PCA, while the embeddings are generated by a highly complex non linear model. Overall, the clustering obtained with MFCCs seems to be better separated compared to the Log Mel Spectrogram ones, even if a decent amount of overlapping is present. The YAMNet embeddings on the other hand, are of difficult interpretation. This could be caused by the linear nature of the PCA, while the embeddings are generated by a highly complex non linear model.

C. KNN

K Nearest Neighbour (KNN) is a simple algorithm that can be used for classification. It builds a lookup table with the training data, and then is able to classify the input, choosing the same label as the majority of the K closest points. K is an input provided by the user. In this project, the euclidean distance is used to measure the closeness, and K is set to 10. Since the algorithm only accepts 2 dimensional structures (samples on the rows and features on the column), the matrices of the 3 audio features extracted early are collapsed in one dimension.

D. Random Forest

Disambiguation: in this sub section, for the word "feature" a generic column of a samples×features matrix is intended, instead of the audio features extracted early. For the same reason of KNN, the matrices of the 3 audio features are collapsed into 1 dimension. Random forest is an ensemble learning method that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. A decision tree is built by iteratively performing the best split of the training set based on a feature, until a certain stopping condition. The best split is the one that reduces heterogeneity or impurity, hence it is computed as the difference between the impurity before the split, and the weighted average of the impurity of the children nodes. For classification tasks, the output of the decision tree is the label of the majority of the elements in the reached leaf node. The Random forest algorithm tries to achieve better results by constructing uncorrelated decision trees with 2 methods:

- Bagging (also called bootstrap aggregating): for each tree, a dataset the same size of the training set is built, by sampling from it uniformly and with replacement.
- Feature selection: at each split, only a subset of the features are considered.

In this project:

- The Gini impurity measure is used.

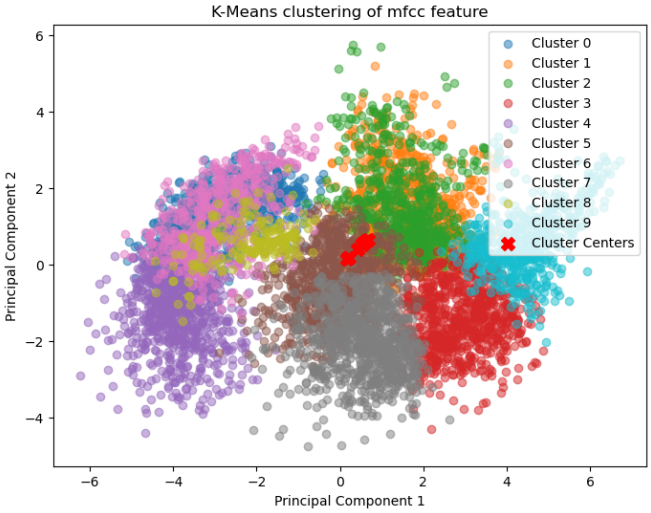

Fig. 3. Clustering with MFCCs

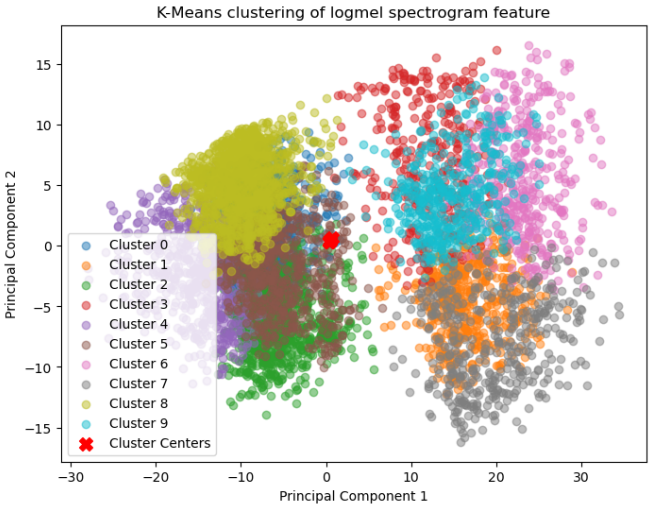

Fig. 4. Clustering with Log Mel Spectrogram

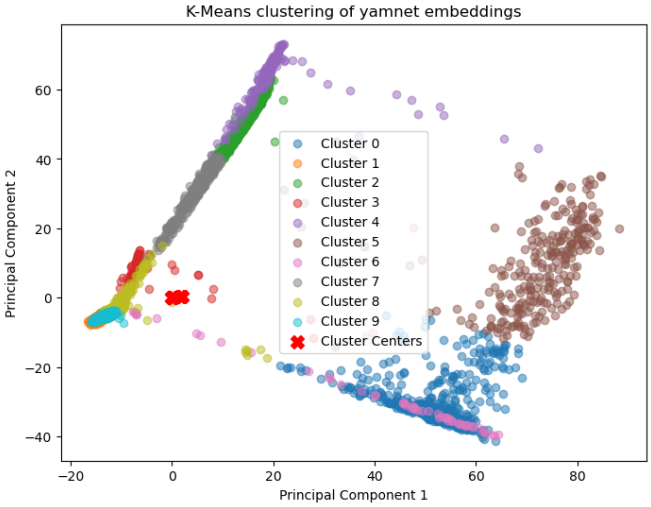

Fig. 5. Clustering with YAMNet embeddings

- 100 decision trees are built.
- At each split, only $\sqrt{n}$ features are considered, where n is the number of features.
- The splitting loop terminates when there is no split that reduces impurity, leading to fully grown trees.

E. MultiLayer perceptron

The MLP is a **feedforward** Artificial Neural Network (ANN), consisting of layers of **fully connected** neurons. Each layer is the input of the next one (**feedforward**), and each neuron outputs the computation of a particular function, called activation function, on the weighted linear combination of its inputs. Each neuron's output is used as input in all the neurons of the next layer (**fully connected**). The first layer is the input layer, where all the feature are used as input of each neuron. All the subsequent layers are called hidden layers, till the output one. Usually, the activation function in the output layer is different from the one used in previous layers. For classification tasks, the output layer can be organized to contain as much neurons as the number of classes, and the activation function usually used is called softmax. This particular function is useful to normalize the output of a network to a probability distribution over predicted output classes. The training is performed by learning the best weights for the network to perform the combinations, in an iterative process where the training data is used as input. Initially the weights are initialized randomly or with other heuristics, then they are adjusted through gradient descent, using a loss function as a reference, the objective being minimizing the error, which can be calculated with the labels of the training set. after the training phase, the model can be used on unseen data, taking the class with the highest probability as the classification result. The YAMNet embeddings are used as input to train a simple MLP, with just one hidden layer composed by 512 neurons, with the Rectified Linear Unit (ReLU) activation function. The output layer contains 10 neurons as the 10 classes, that use the softmax activation function. The model is trained for 100 epochs (iterations) with a callback that stops the training if the loss of the test data doesn't improve for 20 epochs, restoring the best weights. The loss function used is the sparse categorical cross entropy

F. 2D Convolutional Neural Network

A 2D CNN is an ANN designed for processing structured grid data, such as images. It is also popular in the field of audio pattern recognition, since the spectrogram can be viewed as an image. The network is composed of couples of convolutional and pooling layers, till the final classification layer being an MLP with a fully connected hidden layer and an output layer. In convolutional layers, filters (also known as kernels) play a crucial role. Filters are small-sized matrices applied to input data to extract specific features. These filters slide or convolve across the input data, performing element-wise multiplication and summation to produce a feature map. Each filter captures different patterns or features. This structure allows for only partially connected neurons, which share weights (the filter elements). Thus, each neuron computes an element of a specific feature map, and applies an activation function to it. The pooling layer usually follows a convolutional one, and is designed to downsample the spatial dimensions of the feature maps while retaining important information. It does so by taking the max or the average value over a certain number of neurons. The training happens in the same way of the MLP. In this project, 2 2D CNN are built with the same structure and trained on the Log Mel Spectrogram and MFCCs. The network has a convolutional layer consisting of 128 filters with size 3×3 and using the RELU activation function. Then, a max pooling layer is used, with a pool size of 2×2, followed by a 10% dropout layer. Dropout is a technique used during training to prevent overfitting and improve generalization, that works by randomly "dropping" a percentage of the neurons, i.e. setting their activations to 0. Then, a fully connected layer of 512 neurons with relu is used, followed by a 10 neuron output layer with softmax for classification.

G. 1D Convolutional Neural Network

A 1D CNN is like a 2D CNN, but the filters are able to slide only in one dimension. In this project, a 1D CNN is built withe same structure of the 2D one, with filter size of 3 and pooling size of 2, and is trained on the MFCCs matrices, convolving on their heights.

H. Evaluation metrics

To compare the different classifiers, the accuracy on the test set is used, i.e. the number of correct predictions made divided by the total number of predictions made. In addition, to evaluate the goodness of a classifier, other metrics are used. Given a confusion matrix M_{ij} where position ij is the number of samples of class i, classified as class j, the following metrics are used to analyze the classifier with the best accuracy:

- Precision: $Pr_i = \frac{M_{ii}}{\sum_j M_{ji}}$. Basically it is a measure of how accurate the classifier is when it picks a class.
- Recall: $Re_i = \frac{M_{ii}}{\sum_j M_{ij}}$. Basically it measures what percentage of a class is correctly classified
- F1 score: $F_i = \frac{2Re_i Pr_i}{Pr_i + Re_i}$. It is a combination of both precision an recall, giving a unique score of the goodness of the classification, relatively to class i

III. Results

Overall, the MFCCs performed best regardless of the model, as seen in table I. The Log Mel Spectrogram yielded higher accuracy with the Random Forest compared to KNN, probably because it deals better with high dimensionality data.

TABLE I
MODELS AND FEATURES COMPARED BY ACCURACY

	MFCCs	Log Mel Spectrogram	YAMNet embeddings
KNN	0.53	0.34	0.42
Random Forest	0.56	0.54	0.45
MLP			0.52
2D CNN	0.75	0.68	
1D CNN	0.80		

YAMNet embeddings were the worst, even if there was a slight improved with the MLP. A possible explanation could be that YAMNet is trained to recognize 521 sound events, where gunshot is one of the classes. Hence, the extracted embeddings are useful to distinguish the shot from other sounds, but not to tell apart different gunfire types. The CNN architectures performed better, which can be expected since a lot of computer vision methods have shown good results in audio analysis, due to the spectrogram image interpretation. The 1D CNN not only performed better in terms of accuracy (80%), but also was less prone to overfitting, as seen in image 6. In table II the results of the 1D CNN are summarized. The .22LR had the highest f1-score. It was better classified also by the previous models, probably because it is a relatively small caliber compared to the others in the dataset, and thus more distinguishable. As seen in the confusion matrix (figure 7), the 1D CNN confuses .380 and 9mm, the worst performing classes with an f1 score of 0.55 and 0.69 respectively. This could be explained by the bullets having the same diameter and being similar (the 9mm is slightly longer). Also other similar calibers are sometimes misclassified, like .40, .45 and 9mm.

IV. Conclusions

In this work, gun caliber recognition is attempted based on gunshot audio with several models and 3 audio features. The MFCCs performed better than the Log Mel Spectrogram, with an accuracy of 80% used in a simple 1D CNN, while the YAMNet embeddings did not provide the hoped results. While the best model is promising, it needs to be noted that the audio recordings were acquired in a specific and controlled environment, and the dataset contained unbalanced classes with only hundreds to 1 thousand of samples per class. Other works like [2] were table to achieve higher accuracy on larger datasets, employing more complex computer vision based models.

TABLE II
1D CNN RESULTS

	precision	recall	f1-score	support
.223	0.93	0.77	0.84	69
.22LR	0.96	0.93	0.94	273
.308W	0.72	0.97	0.82	68
.357	0.91	0.88	0.90	69
.380	0.72	0.45	0.55	69
.38SPL	0.93	0.84	0.88	67
.40	0.73	0.73	0.73	208
.45	0.76	0.74	0.75	208
7.62x39	0.91	0.79	0.85	66
9mm	0.63	0.76	0.69	206
macro avg	0.82	0.79	0.80	1303
weighted avg	0.80	0.80	0.80	1303

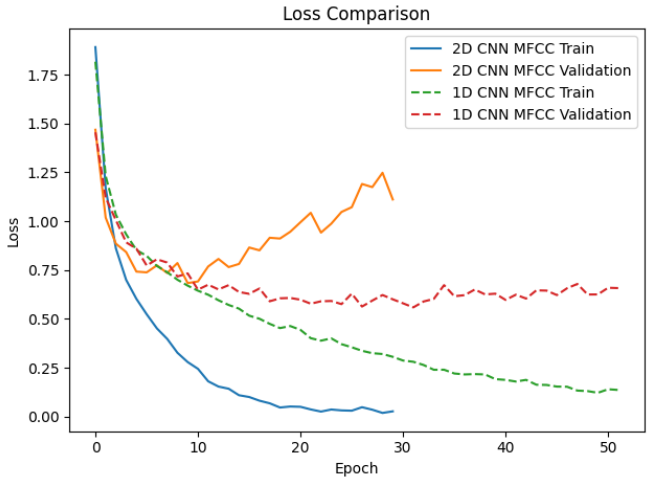

Fig. 6. Loss comparison between 1D and 2D CNN trained on the MFCCs features

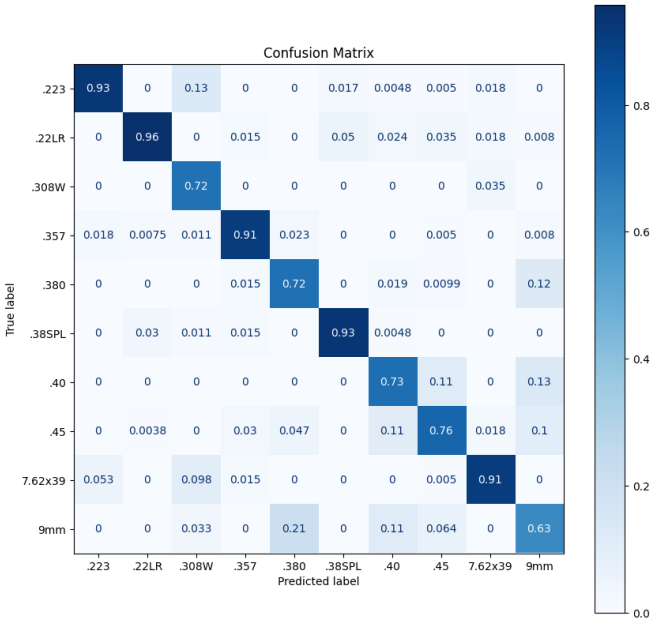

Fig. 7. Confusion matrix of the 1D CNN, normalized on the columns. A percentage p in position ij means that of all the samples classified as class j, p percentage was of class i.

References

[1] R. Nijhawan, S. A. Ansari, S. Kumar, F. Alassery, and S. M. El-Kenawy, "Gun identification from gunshot audios for secure public places using transformer learning," *Scientific reports*, vol. 12, no. 1, p. 13300, 2022.

[2] J. Li, J. Guo, M. Ma, Y. Zeng, C. Li, and J. Xu, "A gunshot recognition method based on multi-scale spectrum shift module," *Electronics*, vol. 11, no. 23, 2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/23/3859

[3] R. Lilien, J. Housman, R. Mettu, T. Weller, L. Haag, and M. Haag, "Gunshot audio forensics dataset," http://cadreforensics.com/audio/, 2023.

[4] M. E. Ylikoski, J. O. Pekkarinen, J. P. Starck, R. J. Pääkkönen, and J. S. Ylikoski, "Physical characteristics of gunfire impulse noise and its attenuation by hearing protectors," *Scandinavian Audiology*, vol. 24, no. 1, pp. 3–11, 1995.