Project

Digital Signal And Image Management

Agazzi Ruben 844736

Sommario—This project is divided in three parts: audio classification were I tried to classify urban sounds into 10 different classes using machine learning algorithms, image classification where I tried to classify images belonging to 10 different classes like ”Forest”, ”Building”, etc... and content based image retrieval on food images from a dataset containing 10 different types of food.

INDICE

1 AUDIO CLASSIFICATION

The first task of the audio consists in the classification of 1-d audio signals into 10 different classes. I solved this task by doing some preprocessing of the dataset, feature extraction and finally classification using the features with a machine learning algorithm.

1.1 Dataset

The dataset can be found on Kaggle, it is called ”UrbanSound8K”[9] and consists in sounds recorded in an urban environment.
The dataset contains 10 different classes:

- Air conditioner
- Car horn
- Children playing
- Dog bark
- Engine Idling
- Gun shot
- Jackhammer
- Siren
- Street music

The dataset contains 8732 labeled sounds, and all the sound have a duration that is less or equal to 4 seconds.

1.2 Preprocessing

After an initial exploratory analysis of the audio data I found out that some audio were mono channel and some were stereo audio. For the stereo audio I took the first channel in order

to process the data in an equal way for all the entries of the dataset.

1.3 Feature extraction

For the phase of feature extraction I decided to extract the following features:

- Energy: is defined as the sum of the squared values of the audio $E = \sum_{n=-\infty}^{\infty} |x(n)|^2$
- Zero Crossing Rate: is defined as the number that the signal switch from a positive to a negative value or vice versa, divided by the length of the signal
- Spectrogram: Is an intensity plot of the fourier transform magnitude. It can be seen as a visual(2-d) representation[3]
- Mel Spectrogram: the Mel spectogram is the spectogram with its value scaled by the Mel scale. The Mel scale was introduced because humans do not easilly tell the difference in the same ways for all the heard frequencies.
- MFCC features: is a representation of the short-term power spectrum of a sound[5].

Some features, like the Mel spectogram, change according to the duration of the audio, but we cannot have features with different sizes for the classification part, so I decided to take the first 1 second for the extraction of these features.

1.4 Classification

After extracting the features I proceeded to the classification task. In this case i trained a Support Vector Machine and a Multi Layer Perceptron in order to compare them.

1.4.1 Support Vector Machines

Support vector machines are a type of machine learning algorithms that tries to classify data by triyng to linearly separate them. Because, in most of the cases, data are not linearly separable, it first applies a transformation to the data, for example it adds some dimensions to the data, in order to make them more easily separable

For the training of the Support Vector Machine I used the python library sklearn, in particular the LinearSVC classifier. I also tried different configurations for the SVC parameters:

- C parameter: Is a regularization parameter, and the strenght of the regularization is inversely proportianl to its value. In this case the best parameter found was

After training the classifier I tried it in the test dataset in order to evaluate it's performance. The accuracy of the classifier was equal to 69% and the following values abount precision, recall and f1 on the 10 classes:

Class	Precision	Recall	F1
0	0.91	0.84	0.88
1	1.00	0.31	0.47
2	0.69	0.58	0.63
3	0.30	0.83	0.44
4	0.84	0.64	0.73
5	0.93	0.83	0.87
6	1.00	0.33	0.50
7	0.96	0.85	0.90
8	0.94	0.67	0.78
9	0.58	0.54	0.56

Tabella 1: Table of Precision, Recall and F1 measure on every class computed on the test dataset

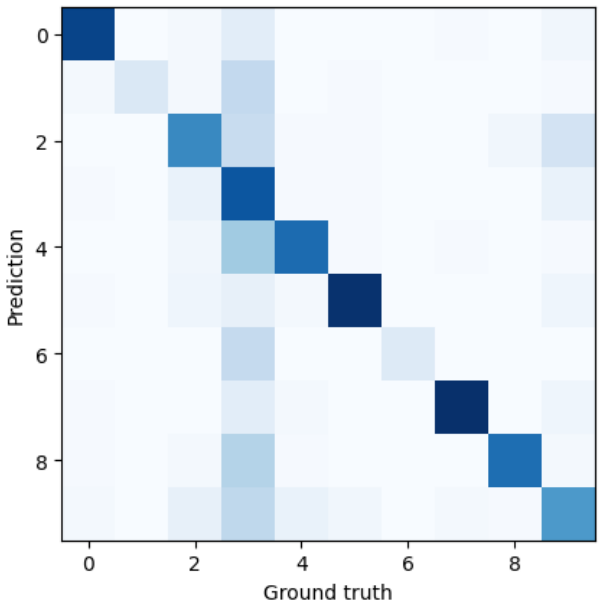

Figura 1: Confusion matrix of the classifier over the test dataset

1.4.2 Convolutional Neural Network

The convolutional neural network composed by 11 layers. I also applied L2 regularization to the layers in order to prevent overfitting. The layers are the following:

- First convolutional layer: Convolutional layer with 128 filters of size 3x3 with ReLU activation
- First max pooling layer: max pooling layer with window size of 2x2
- First flatten layer: flatten layer in order to pass the features to the dense layers
- First dense layer: Dense layer with 1024 neurons with ReLU activation
- Second dense layer: Dense layer with 512 neurons with ReLU activation
- Thirs dense layer: Dense layer with 256 neurons with ReLU activation
- Fourth dense layer: Dense layer with 128 neurons with ReLU activation
- First dropout: Dropout layer with probability of dropout of 0.2 to add regularization
- Fifth dense layer: Dense layer with 64 neurons with ReLU activation
- Second dropout: Dropout layer with probability of dropout of 0.1 to add regularization
- Output layer: Dense layer with 10 neurons with softmax activation to perform classification

The model reached an accuracy of 59.8% on the validation set and an accuracy of 60% on the test dataset. The model obtrained the followed precision, recall and F1 measure over the 10 classes:

Class	Precision	Recall	F1
0	0.88	0.52	0.65
1	0.85	0.67	0.75
2	0.66	0.31	0.42
3	0.67	0.58	0.62
4	0.83	0.66	0.74
5	0.28	0.91	0.43
6	0.96	0.70	0.81
7	0.83	0.64	0.72
8	0.85	0.61	0.71
9	0.75	0.45	0.56

Tabella 2: Table of Precision, Recall and F1 measure on every class computed on the test dataset

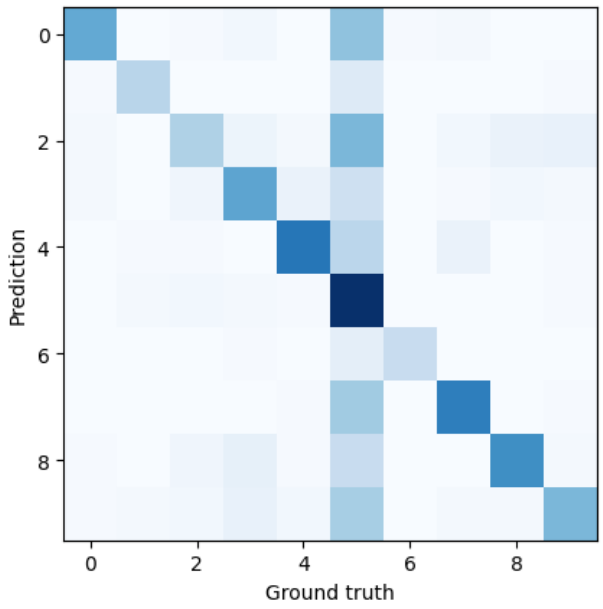

Figura 2: Confusion matrix of the classifier over the test dataset

1.5 Evaluation

The best performing model was the smv. This could be because for the convolutional neural network I only used MFCC features for classification. Maybe adding more features can lead to better result using this type of architecture. Also I observed making tests on downloaded data that both the models tends to predict always the same class, maybe there is some kind of overfitting over the dataset itself. This could be maybe fixed by using a larger dataset or by using better features for classification.

2 IMAGE CLASSIFICATION

For the image classification I tried to clasify images beloging to 6 different classes using a custom Convolutional Neural Network and using a pretrained Convolutional Neural Network.

2.1 Dataset

The dataset is the Intel Image Classification[4] available in Kaggle, it consists in 25000 rgb images of size 150x150 divided in 6 classes. The classes are:

- Buildings
- Glacier
- Mountain
- Sea
- Street

2.2 Convolutional Neural Networks

Convolutional Neural Networrks are a class or artificial neural networks, most commonly used to analyze images[1]. They are deep learning algorithms presenting convolutional layers: theese layers are composed by a fixed number of convolutional filters of a given dimension. The network during the training phase try to learn the weight of the components of the convolutional filters

2.3 Custom CNN

For the classification problem I designed a custom CNN that I later trained using the training data. The Network is composed by 14 layers:

- Input layer: Input layer for images of size 150x150x3
- First Convolutional layer: Convolutional layer composed by 64 convolutional filters of size 3x3 with ReLU activation function.
- First average pooling: Average pooling layer with filter size 2x2
- Second Convolutional layer: Convolutional layer composed by 32 convolutional filters of size 3x3 with ReLU activation function.
- Second average pooling: Average pooling layer with filter size 2x2
- Third Convolutional layer: Convolutional layer composed by 16 convolutional filters of size 3x3 with ReLU activation function.
- Third average pooling: Average pooling layer with filter size 2x2
- First flatten layer: layer used to flatten the output of convolutional layers in order to have a vector of features to pass to the fully connected part of the network
- First dense layer: Dense layer with 512 neurons with ReLU activation function
- First dropout layer: Dropout layer with probability of 0.1 to prevent overfitting
- Second dense layer: Dense layer with 256 neurons with ReLU activation function
- Second dropout layer: Dropout layer with probability of 0.1 to prevent overfitting
- Third dense layer: Dense layer with 64 neurons with ReLU activation function
- Output layer: Dense layer with 6 neurons and softmax activation function in order to make the classification

The final network has a total of 2,539,894 trainable parameters. I trained the network on the training dataset for 128 epochs obtaining an accuracy on the validation set equal to 76% but an accurazy of 64% on the test dataset.

Class	Precision	Recall	F1
0	0.65	0.59	0.62
1	0.81	0.86	0.83
2	0.57	0.58	0.58
3	0.54	0.67	0.60
4	0.57	0.43	0.49
5	0.73	0.71	0.72

Tabella 3: Table of Precision, Recall and F1 measure on every class computed on the test dataset

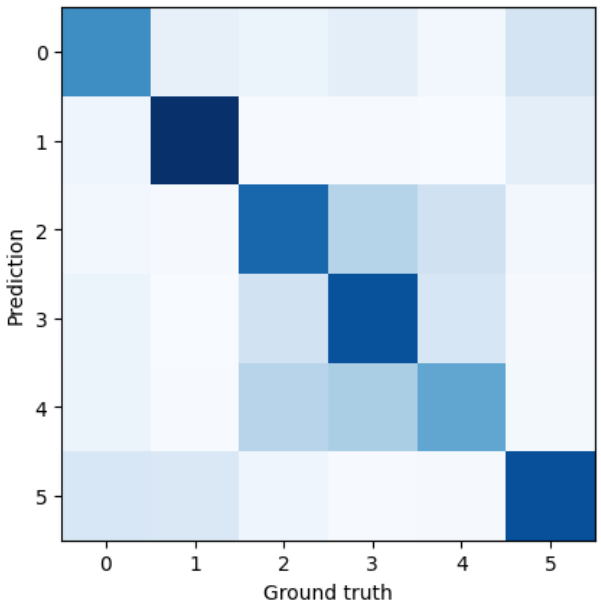

Figura 3: Confusion matrix of the classifier over the test dataset

Class	Precision	Recall	F1
0	0.85	0.92	0.89
1	0.97	0.94	0.95
2	0.68	0.84	0.75
3	0.88	0.46	0.60
4	0.69	0.85	0.76
5	0.89	0.88	0.89

Tabella 4: Table of Precision, Recall and F1 measure on every class computed on the test dataset

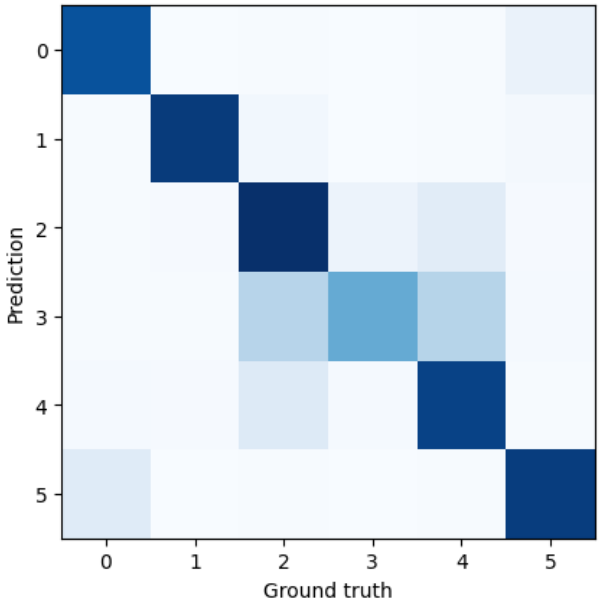

Figura 4: Confusion matrix of the classifier over the test dataset

2.4 Pretrained CNN

In this other try i used a pretrained CNN, the VGG16[6] network using the ImageNet weights. I fine tuned the network in order to classify the data: I removed the initial input layer because the input size was greater than the dimension of the images of the dataset, and then I added 4 Dense layers at the end of the network that I trained to classify the images, in practice I use all the VGG16 network as a feature extractor. With this model I obtained an accuracy of 83.06% on the validation dataset and of 81% over the test set. For the training of this model I also added a step of data augmentation in order to generate more images by sheering, rotating, zoomming and changing the brightness of the original images. The classifier obtained the following metrics over the 6 classes:

2.5 Evaluation

After evaluating the two classifier the best one seems to be the fine tuned VGG16 model, because all the measures were higher than the convolutional neural network; but testing on some downloaded images from the internet, I saw that the fine tuned model always predicted the same class, while the other classifier predicted better classes for the images, even though it had problems like distinguishing a forest from a mountain if the forest had some altures, or distinguishing between a building and a street.

3 Image Retrieval

For the part of image retrieval I used the same pretrained network used in the previous task,

the VGG16 model. For this phase I made two tests: I used features for the retrieval coming from the third pooling layer and from the last pooling layer.

3.1 Dataset

The dataset is available on kaggle and is the ”Fast Food Classification Dataset - V2”[2]. It contains 20.000 images belonging to 10 different classes:

- Baked potato
- Burger
- Crispy Chicken
- Donut
- Fries
- Hot Dog
- Pizza
- Sandwich
- Taco
- Taquito

3.2 Feature extraction

For the phase of feature extraction I used the pre-trained Concolutional Neural Network VGG16 and cutted the model at the third pooling layer. After extracting the feature i flatten them and finally save them in order to be used to find similar images. Teh final otained features have a dimansionality of 25088.

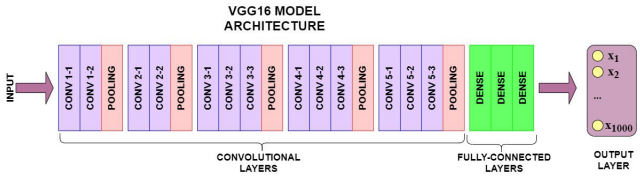

Figura 5: VGG16 model architecture

3.3 Retrieval

For the image retrieval task I used a k-d tree[8]. A kd-tree is a structure, in particular a binary tree, used for storing a finite set of points form a k-dimensional space. It partitions the k-dimensional space in order to organize the points. This can lead to faster search performance of the point inside the k-dimensional space. In this case I used the ”pynanoflann”[7] python lybrary, because was the only library I tried that could build the tree in reasonable time(a few seconds). To retrieve the images starting from a test image we need to use the cutted VGG16 model to extract the features and then search for the most similar features inside the tree. The distance function used for searching the features inside the tree was the euclidean distance. The tree returns the 10 most similar features with the corresponding index.

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + ... + (p_n - q_n)^2}$$

Figura 6: Euclidean distance equation

3.4 Performance evaluation

In order to evaluate the performance of the information retrieval system, I decided to take the test dataset, already partitioned, and use it to check how many retrieved images belongs to the same class of the tested image. The test set is composed of 1500 images belonging to the 10 different classes. After retrieving the most similar images I calculated the accuracy of the class of the retrieved 15000 images compared to the class of the test image. The system has an overall accuracy of 19%.

4 Conclusions

After making and evaluating all the 3 systems I made the following conclusions based on the results.

4.1 Audio classification

Between the two tested classifiers the best performing one was the Support Vector Machine which used several features for classification. Maybe adding more features to the neural network could have reached better performance, considering that with only MFCC features have achieved a close level of accuracy to the SVM.

4.2 Image classificaiton

The best performing model is the fine tuned VGG16 network. The custom convolutional neural network has very low accuracy on the test set but testing on downloaded samples it seems to perform better than the fine tuned model, that in most of the cases predicts always the same class.

4.3 Image retrieval

The image retrieval system achieved discrete results, even though on the test set the accuracy of the images retrieved, compared by class, is pretty low(18%). Even though, after testing I observed that usually, out of the 10 found images,

RIFERIMENTI BIBLIOGRAFICI

[1] Dan C. Ciresan et al. *Flexible, High Performance Convolutional Neural Networks for Image Classification*. URL: https://people.idsia.ch/~juergen/ijcai2011.pdf.

[2] *Fast Food Classification Dataset - V2*. URL: https://www.kaggle.com/datasets/utkarshsaxenadn/fast-food-classification-dataset.

[3] Julius O. Smith III. *Mathematics of the Discrete Fourier Transform (DFT), with Audio Applications — Second Edition*. URL: https://ccrma.stanford.edu/~jos/mdft/Spectrograms.html.

[4] *Intel Image Classification*. URL: https://www.kaggle.com/datasets/puneet6060/intel-image-classification.

[5] *Mel-frequency cepstrum*. URL: https://en.wikipedia.org/wiki/Mel-frequency_cepstrum.

[6] Karen Simonyan e Andrew Zisserman. *VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION*. URL: https://arxiv.org/pdf/1409.1556.pdf.

[7] u1234x1234. *pynanoflann*. URL: https://github.com/u1234x1234/pynanoflann.

[8] Andrew W. Moore Carnegie Mellon University. *An into ductory tutorial on kd-trees*. URL: https://www.ri.cmu.edu/pub_files/pub1/moore_andrew_1991_1/moore_andrew_1991_1.pdf.

[9] *UrbanSound8K*. URL: https://www.kaggle.com/datasets/chrisfilo/urbansound8k.