**PRESENTATION SCRIPT**

Good morning, we are Luca Masiero and Stefano Ivancich

**Outline**

Today, we are going to talk about the Keyword Spotting problem, present a possible solution for it, discuss about the 3 architecture we made and we will compare them against other papers.

**The problem**

The goal of Keyword Spotting is to detect a relatively small set of predefined keywords in a stream of user utterances, usually in the context of an intelligent agent on a mobile phone, a consumer “smart home” device or a robotics application.

It is desirable that such models have a small footprint (for example, measured by the number of model parameters) and a fast prediction speed so they can be deployed on low-power and performance limited devices.

**The Solution**

One of the main challenges of neural networks in audio processing is that the length of the input sample should be fixed, but the sound captured may have various duration.

The aim of the end-to-end architecture we propose, represented in this figure, is to handle audio signals of variable lengths.

One way to avoid the constraint imposed by the input layer of the neural network, is to split the audio signal into different frames of fixed length using a sliding window of 1-second width.

However, letting the model continuously process each frame is too computational expensive. For this reason, we can use a VAD (Voice Activity Detection) or a Silence Filter to let the model process just the frames containing voice.

This component must be very compact, low computational expensive and very fast, it could be part of a software or even a little piece of hardware.

Only a batch of subsequent windows will be processed. For each of these windows a Feature Extraction module will extract 80 Mels, 40 MFCC, 40MFCC + s or no extraction at all: this will be the input to our model.

During the classification we need to aggregate the various predictions to come up to a single decision. For this reason, different fusion rules can be used to reach a final decision, such as the majority vote or the sum rule.

**What we tried**

We used the Google Speech **Dataset** V2. This dataset contains 105,829 audio files of 1 second divided in thirty classes.

We decided to create two different datasets: the first one is made of ten classes, the second is made of 21 classes.

We decided to study four different types of **features** in order to compare them and tried to see which was the best able to solve our problem:

* No feature extraction: we trained a specific model (1DCNN) directly on the raw signal (a 16,000 elements vector)
* 80 Mels spectrogram:
* 40 Mel-frequency cepstral1 coefficients (MFCC):
* 40 MFCC with their first and second derivatives.

We tried different type of **neural networks**: 1D Cnn on raw data and DSConv Cnn on features and build an ensemble of the best models.

Furthermore we implemented different model sizes in order to fit different devices.

**1D CNN on RAW data**

The proposed architecture, shown in Figure 2, is made of four convolutional layers interlaced with max pooling layers and followed by two fully connected layers and an output layer.

The input consists of an array of 16,000 dimensions, which represents a 1-second audio sample at 16kHz.

The output of the last pooling layer, for all feature maps, is flattened and represents the input for a fully connected layer. In order to reduce overfitting, batch normalization is applied after the activation function of each convolution layer. After the last pooling layer there are two fully connected layers with 128 and 64 neurons respectively; dropout is applied with a probability of 0.25 for both layers.

RESULTS DISCUSSIONNNNN………………………….

**Separable CONV**

**Separable convolution:** uses less parameters, less memory and less computations than regular convolutional layers, and perform better.

There are two types of Separable convolution: Spatial and depthwise

**Spatial separable convolution:** divides a kernel into two, smaller kernels.

**Problem**: not all kernels can be “separated” (mathematically) into two.

**Depthwise Separable convolution:** uses kernels that cannot be “factored” into two smaller kernels. It deals not just with the spatial dimensions, but also with the depth dimension (number of channels).

Splits a kernel into 2 separate kernels that do two convolutions:

* the depthwise convolution
* the pointwise (1x1) convolution.

**Figure Separable CONV**

3D ….

Here we have the Depth Separable convolution layer that we implemented. It is composed of DSConv2D and a pointwise 2Dconvolution, with batch normalization and Relu activation.

**DSConv Model - Large**

The model take as input the Mel spectrum or MFCC or MFCC+ deltas.

Is made of a normal 2D CNN followed by batch normalization and 5 DSConv2D layers. An average pooling layer followed by a fully-connected layer, is used in the end to provide global interaction and reduce the total number of parameters in the final layer.

We implemented three different variants: Large, Medium and Small, where we tried to reduce the number of parameters still maintaining a good accuracy.

RESULTS DISCUSSIONNNNN………………………….

**DSConv Model - Medium**

The network is the same has before but with a DSConv layer less and, for each confolution we use less filters. For example in the large we used 276 filters while in the medium 172.

RESULTS DISCUSSIONNNNN………………………….

**DSConv Model - Medium**

As before we reduced the number of filters of each convolution to 64.

RESULTS DISCUSSIONNNNN………………………….

**Ensemble 10-cmd e 21-cmd**

As stated before, we built three kinds of ensembles: Large, Medium and Small, in order to have models that can fit different hardware sizes. The ensembles simply pick the 3 best models and compute the average of their final prediction, in this way we reached the best accuracy paying a high price: larger networks.

RESULTS DISCUSSIONNNNN………………………….

**Performances 10-cmd**

RESULTS DISCUSSIONNNNN………………………….

**Performances 21-cmd**

RESULTS DISCUSSIONNNNN………………………….

**Conclusions**

**Conclusions**

* The tests showed that our model was **very good at classify keywords**.
* Unfortunately we didn’t beat the state-of-the-art models.
* We found that the number of convolutional layers played a key role in detecting high-level concepts
* There is not so much difference between using 80 Mels or 40 MFCCs.
* We presented different model sizes in order to fit different devices.

**Future Work:**

* Try different hyper-parameters during training
* Change the **structure** of the **network** using:
  + SincConv
  + GDSConv
* Build a Feature extractor with convolutional autoencoder

**Live DEMO**

…………….