Deep Learning and Natural Language Processing - Acoustic Features

Vincenzo Scotti,
Ph.D. student
vincenzo.scotti@polimi.it

arcslab
adaptable, relational and cognitive software environments

N.L.P. – A.Y. 20-21

Contents

This symbol means that the formulae in the slide aren't required to be known

PRE-DEFINED FEATURES

Acoustic features

- To work on acoustic signals it is advisable to use high level features as input as it happens for text
- Differently from text, audio in general and voice in particular can count on a wide variety of pre-defined features of different level of complexity
  - One can select from this pool those that consider more appropriate to the specific task
- These features are usually extracted from a sliding window over the input signal

Acoustic features (taxonomy)

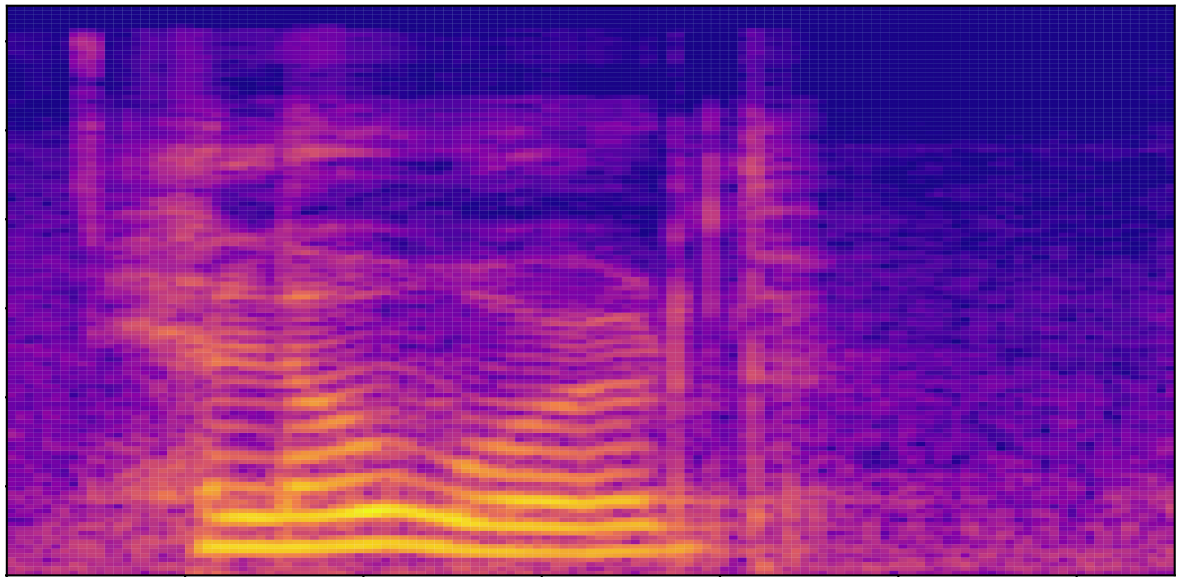
- Acoustic features are divided into **temporal** and **spectral**
- Temporal features include those that are extracted solely from the time domain
- Spectral features are those extracted from the frequency domain

Temporal vs. spectral features

- **Temporal** Features:
  - Pitch
  - Intensity
  - Harmonicity
  - Jitter
  - Shimmer
  - Speech rate
  - Short term energy
  - Sort term entropy of energy
  - silences
  - ...
- **Spectral** Features:
  - (Mel) Spectrogram
  - MFCC
  - Spectral centroid
  - Spectral spread
  - Spectral entropy
  - Spectral flux
  - Spectral roll-out
  - Chroma
  - ...

Spectrogram

- Example of a spectrogram
  - x-axis is time
  - y-axis are the frequency bins
  - Intensity is represented through the color



Usage of pre-defined features

- These features can be used as sequential input for neural networks
  - They can be stacked into a 2D tensor
- Provided that they rely on the same time axis, features sequences can be concatenated along the channel axis
- A neural network can be trained on top of these features
- Particularly useful when there aren't enough samples in the data set to train a deep model

DEEP FEATURES

Transferring deep features

- Sometimes pre-defined features are not sufficient to obtain good performances from a neural network
- Sometimes the data set is not sufficiently big to train a deep neural network to learn good features from raw input
- Resort to transfer learning and fine-tuning
  - Instead of selecting manually features, re-use those from a pre-trained model ove a bigger task

Available models

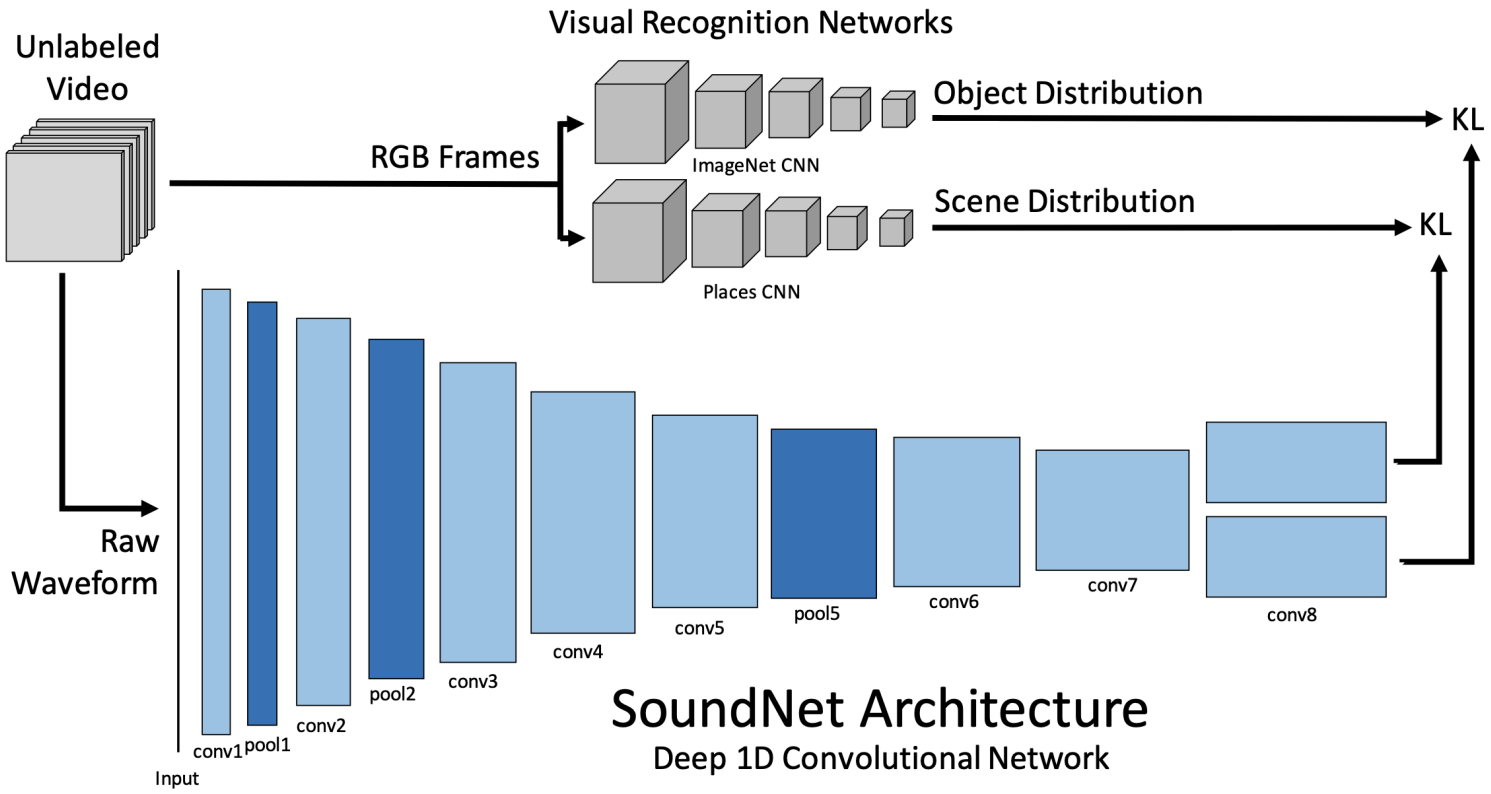
- Through the last years a few models have been developed for this purpose
- Mainly two approaches:
  - Learn from raw input signal
  - Learn from spectral representation

SoundNet

- SoundNet is a CNN working on raw audio signal
- It's trained in a completely unsupervised manner
  - Training task is to replicate the labelling produced by visual analysis CNNs (object and scene identification) taking audio as input
  - It uses KL as loss function
- The pertained weights can be easily transferred to many other tasks

SoundNet

- SoundNet architecture



VGGish

- Another approach that is becoming popular nowadays is that of re-using CNN architectures for image classification to train audio classifiers on very huge data sets
  - The side effect is to produce very general features
- The problem is that visual CNNs expect as input an image (a 3D tensor or 2D feature map)
- To provide a compatible input is possible to threat the spectrogram as grayscale image
  - Spectrogram is a 2D tensor of shape (time steps $\times$ frequency bins), it can be reshaped inserting a dummy axis of 1 dimension in order to have (time steps $\times$ frequency bins $\times$ 1)

VGGish

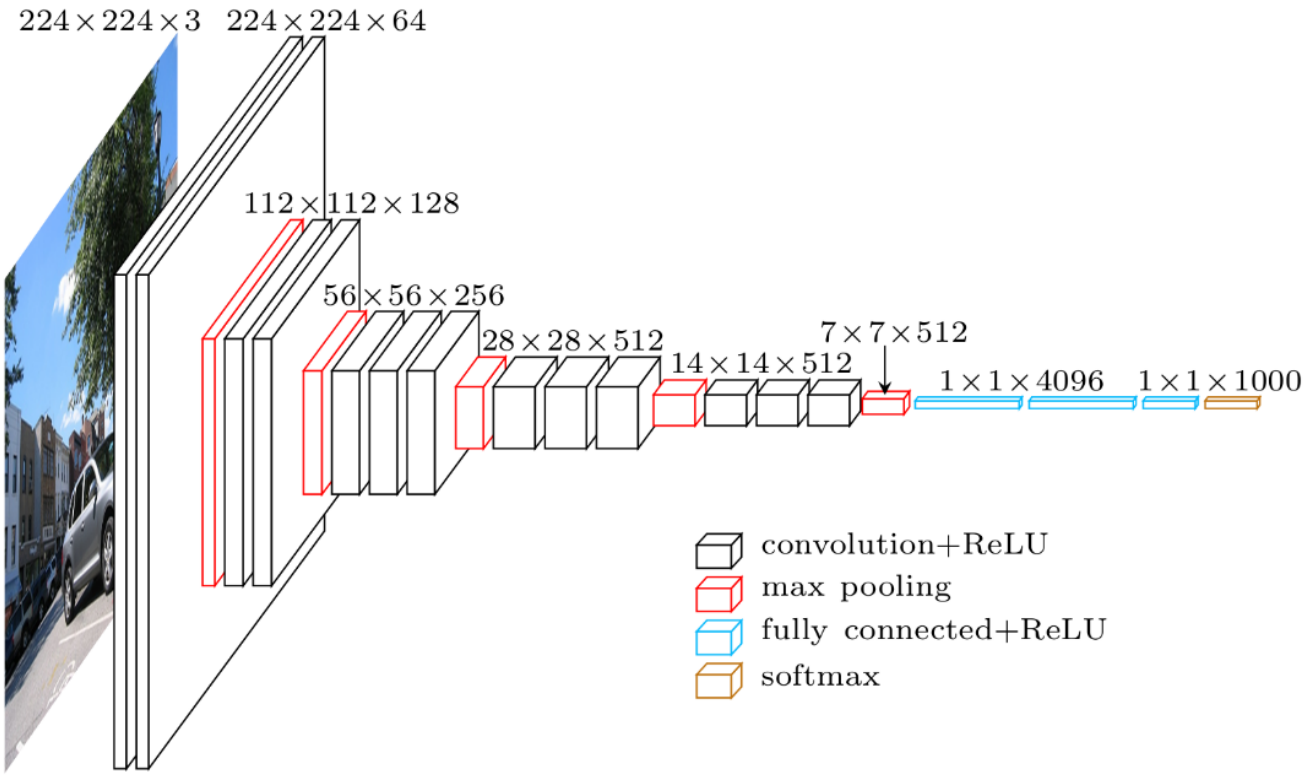
- Once the input is compatible with the CNN input, the convolution kernels can be slid over the feature map to produce the hidden representation
  - The hidden representation is still a 3D tensor
- To get back to a representation compatible with time series data the last 2 dimensions of the feature map should be flattened
  - It is necessary to use this network for feature extraction
  - Just unroll into a feature vector the content of each time step
    (time steps $\times$ frequency bins $\times$ hidden features) $\rightarrow$
    (time steps $\times$ (frequency bins $\cdot$ hidden features))

VGGish

- The network is trained to classify a huge audio data set
  - Audio set is composed of 1M samples of labelled acoustic events
- VGGish is just an example of this approach
  - It leverages the architecture of VGG, a CNN for image classification
- Other architectures have been used with success on this same task
  - Xception
  - ResNet
    - There exist also a variant using ResNet with a special pooling layer called GhostVLAD that velds particularly good features
  - ...

VGGish

- VGG original image classification architecture



Wav2Vec

- Up to now we have seen transferrable deep features for generic audio inputs
- We may be interested in something more specific for voice analysis
- Wav2Vec proposes a generic solution that is trained specifically on voice signal
  - Helped improve ASR task

Wav2Vec

- The idea is to obtain good features though self-supervised predictive pre-training
  - Similar to decoder approach for contextual embeddings
- Training is split in two parts
  - Learn **auto-encoder** on raw input
  - Learn predictive model from auto-encoder hidden representation

Wav2Vec

- Convolutional auto-encoder
  - Train a CNN from raw input signal window $\mathbf{X}_{(k)} \in \mathbb{R}^{(t \times 1)}$ that performs two steps (the k-th window):
    - Project the input signal window into a "compressed" hidden representation $\mathbf{z}_{(k)} = f_{\text{encoder}}(\mathbf{X}_{(k)}) \in \mathbb{R}^{t' \times h}$ with $t' \cdot h < t \cdot 1$ (i.e. **encode** it)
    - Back-project hidden representation in the original space $\mathbf{X}'_{(k)} = f_{\text{decoder}}(f_{\text{encoder}}(\mathbf{X}_{(k)})) \in \mathbb{R}^{t \times 1}$
  - The objective of this training is to minimize reconstruction error

Wav2Vec

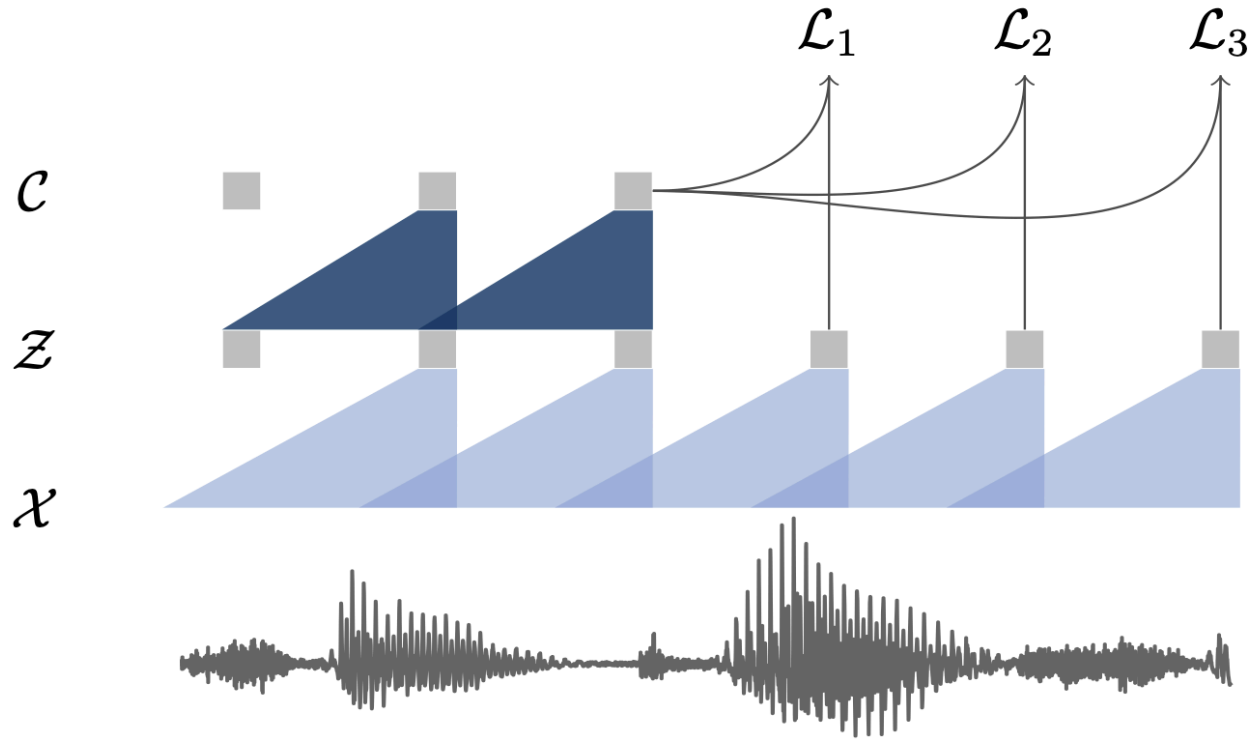
- Predictive causal convolutions
  - Train a CNN on the pre-trained encoder input to predict encoded representations
    - In practice learns an auto-regressive moving average model to predict a sample s steps ahead $\mathbf{z}'_{(k+s)} = h_{(s)}(\mathbf{c}_{(k)}) = h_{(s)}(g(\mathbf{z}_{(k-v:k)}))$ from a window of v elements up to k
    - Cut-off the prediction layers to extract a feature map useful to train another network $\mathbf{C}$ such that $\mathbf{C}[k] = \mathbf{c}_{(k)}$
    - The network trains multiple output layers for varying length predictions
  - The objective of this training is to minimize the contrastive loss (positive/negative sampling) over the predictions at each step s

    $$E_s(\vartheta) = -\sum_k \left(\log \sigma(\mathbf{z}^\top_{(k+s)} \cdot \mathbf{z}'_{(k+s)}) + \lambda \mathbb{E}\left[\log \sigma(-\tilde{\mathbf{z}}^\top_{(k+s)} \cdot \mathbf{z}'_{(k+s)}) \right] \right)$$

    where $\tilde{\mathbf{z}}$ is a negative sample and λ is the weight for negative samples in the loss

Wav2Vec

- Wav2Vec visualized



REFERENCES

References

- https://arxiv.org/pdf/1610.09001.pdf
- https://towardsdatascience.com/extract-features-visualize-filters-and-feature-maps-in-vgg16-and-vgg19-cnn-models-d2da6333edd0
- https://arxiv.org/pdf/1609.09430.pdf
- https://static.googleusercontent.com/media/research.google.com/it//pubs/archive/45857.pdf
- https://arxiv.org/pdf/1902.10107.pdf
- https://arxiv.org/pdf/1904.05862.pdf