AUDIO FEATURES & MACHINE LEARNING

E.M. Bakker

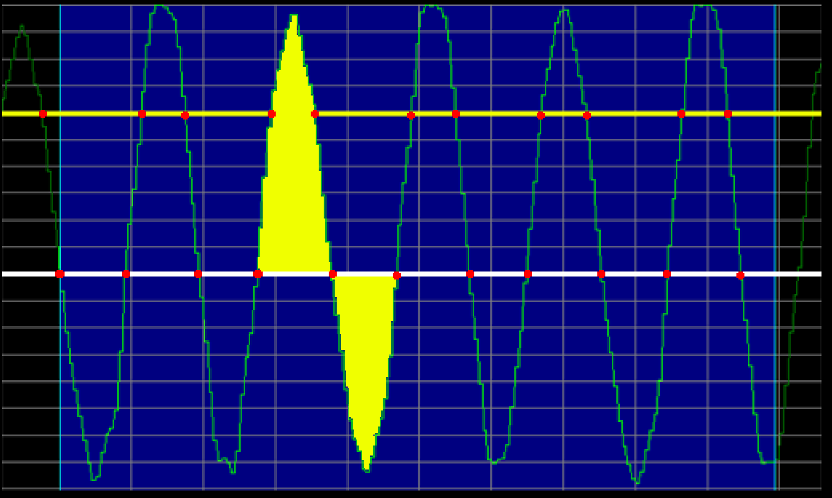
API2022

FEATURES FOR SPEECH RECOGNITION AND AUDIO INDEXING

- Parametric Representations
  - Short Time Energy
  - Zero Crossing Rates
  - Level Crossing Rates
  - Short Time Spectral Envelope
- Spectral Analysis
  - Filter Design
  - Filter Bank Spectral Analysis Model
  - Linear Predictive Coding (LPC)
  - MFCCs

FEATURES FOR SPEECH RECOGNITION AND AUDIO INDEXING

- Parametric Representations
  - Short Time Energy
  - Zero Crossing Rates
  - Level Crossing Rates



Example: Speech of length 0.01 sec.

FEATURES FOR SPEECH RECOGNITION AND AUDIO INDEXING

- Spectral Analysis
  - Fourier Transform
  - Filter Design
  - Filter Bank Spectral Analysis Model
  - Linear Predictive Coding (LPC)
  - MFCCs

SHORT TIME FOURIER TRANSFORM SHORT HAMMING WINDOW: 50 SAMPLES (=5MSEC)

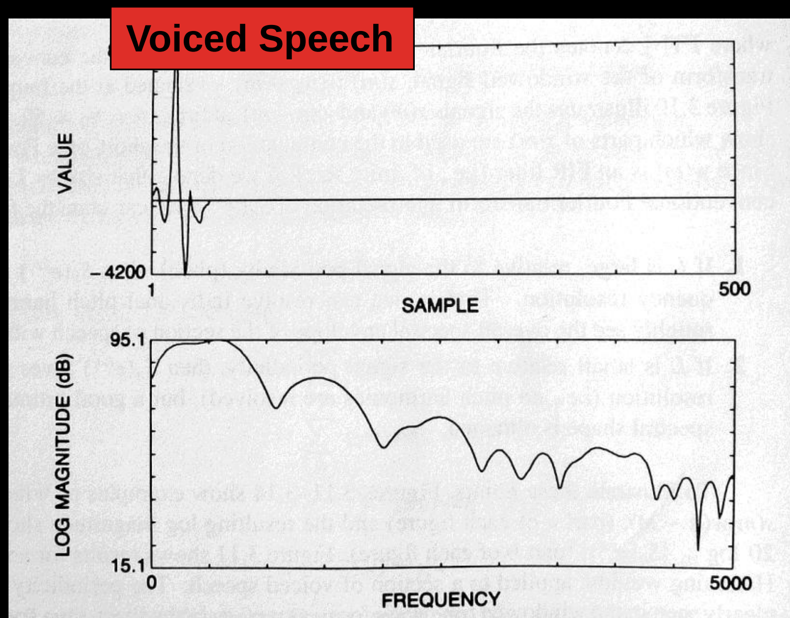


Figure 3.12 Short-time Fourier transform using a short (50 points or 5 msec) Hamming window on a section of voiced speech.

From: Rabiner et al.

SHORT TIME FOURIER TRANSFORM LONG HAMMING WINDOW: 500 SAMPLES (=50MSEC)

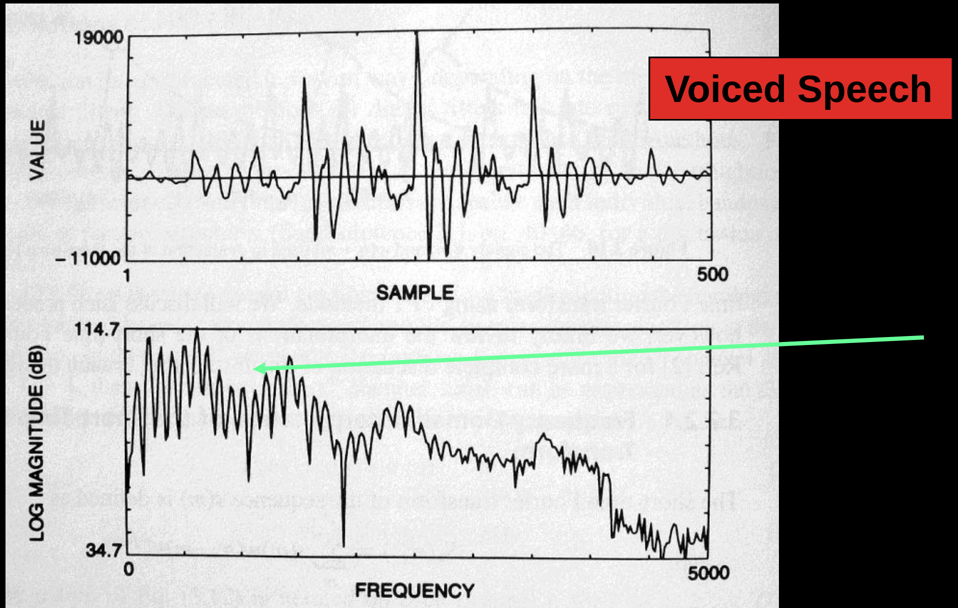


Figure 3.11 Short-time Fourier transform using a long (500 points or 50 msec) Hamming window on a section of voiced speech.

From: Rabiner et al.

SHORT TIME FOURIER TRANSFORM SHORT HAMMING WINDOW: 50 SAMPLES (=5MSEC)

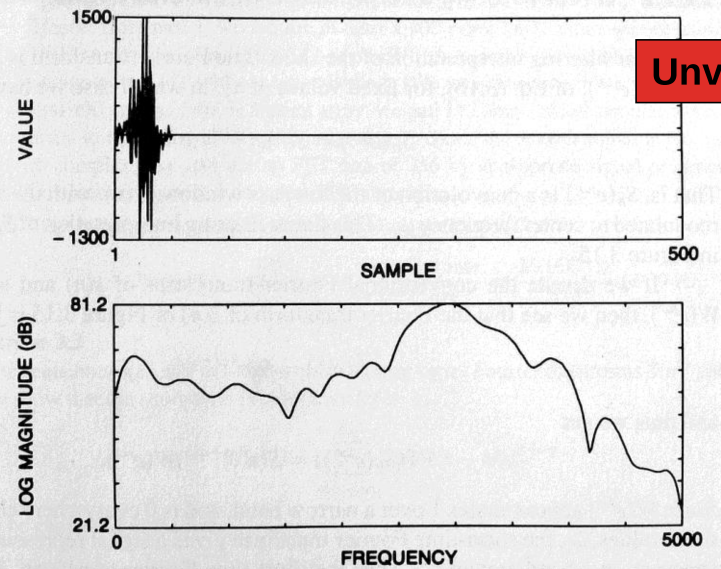


Figure 3.14 Short-time Fourier transform using a short (50 points or 5 msec) Hamming window on a section of unvoiced speech.

Unvoiced Speech

From: Rabiner et al.

SHORT TIME FOURIER TRANSFORM LONG HAMMING WINDOW: 500 SAMPLES (=50MSEC)

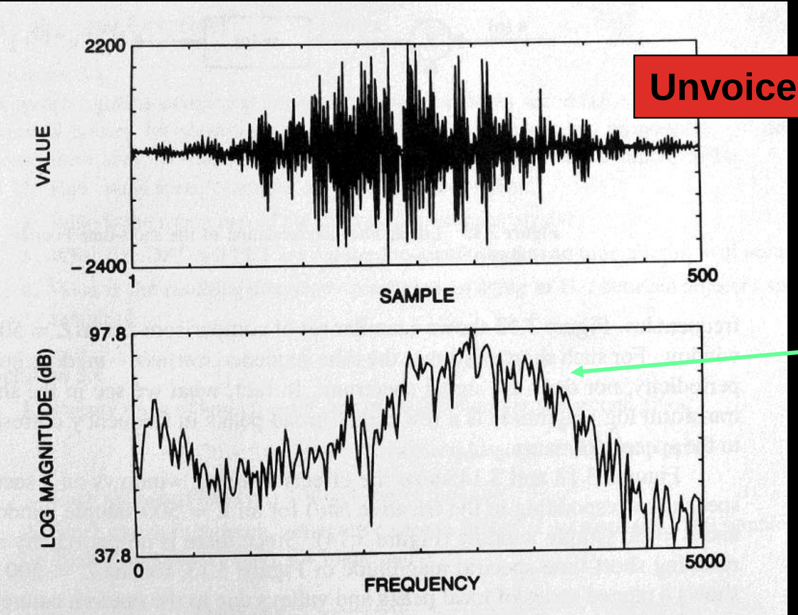
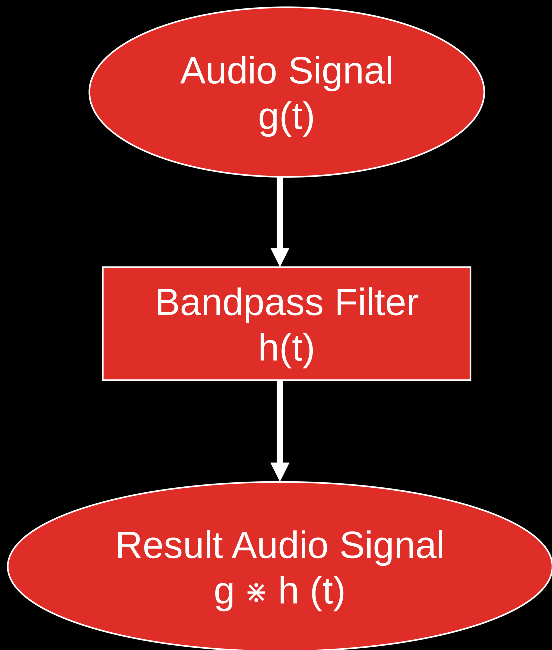


Figure 3.13 Short-time Fourier transform using a long (500 points or 50 msec) Hamming window on a section of unvoiced speech.

From: Rabiner et al.



BAND PASS FILTER

Note that the band pass filter can be defined as:

- a *convolution* with a filter response function h(t) in the time domain
- a *multiplication* with a filter response H(f) function in the frequency domain

$$g * h\ (t) = \int_{-\infty}^{\infty} g(\tau)h(t-\tau)d\tau \leftrightarrow G(f) \cdot H(f)$$

BANK OF FILTERS ANALYSIS MODEL

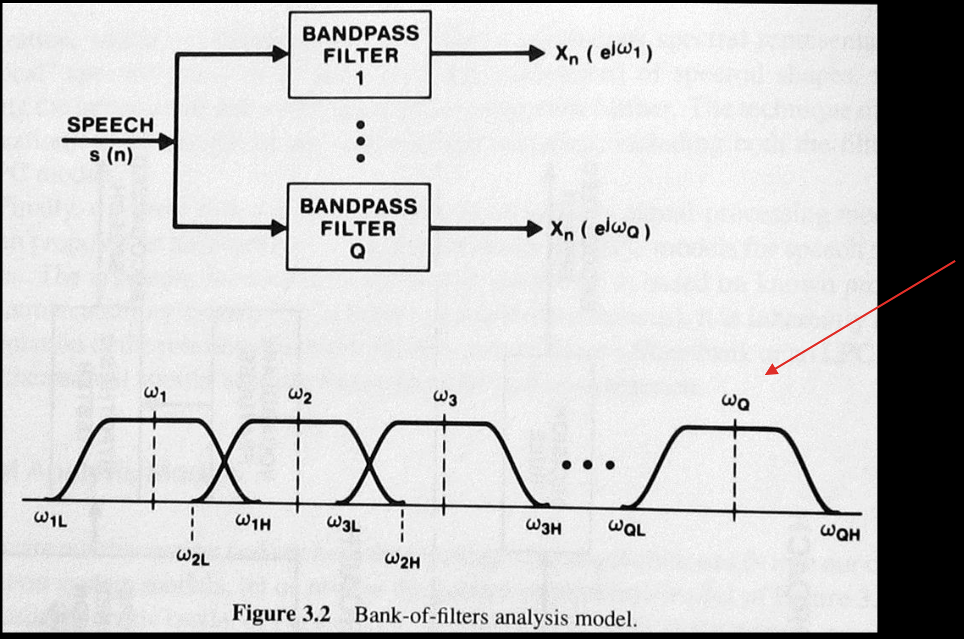


Figure 3.2 Bank-of-filters analysis model.

MEL-CEPSTRUM [4]

Auditory characteristics

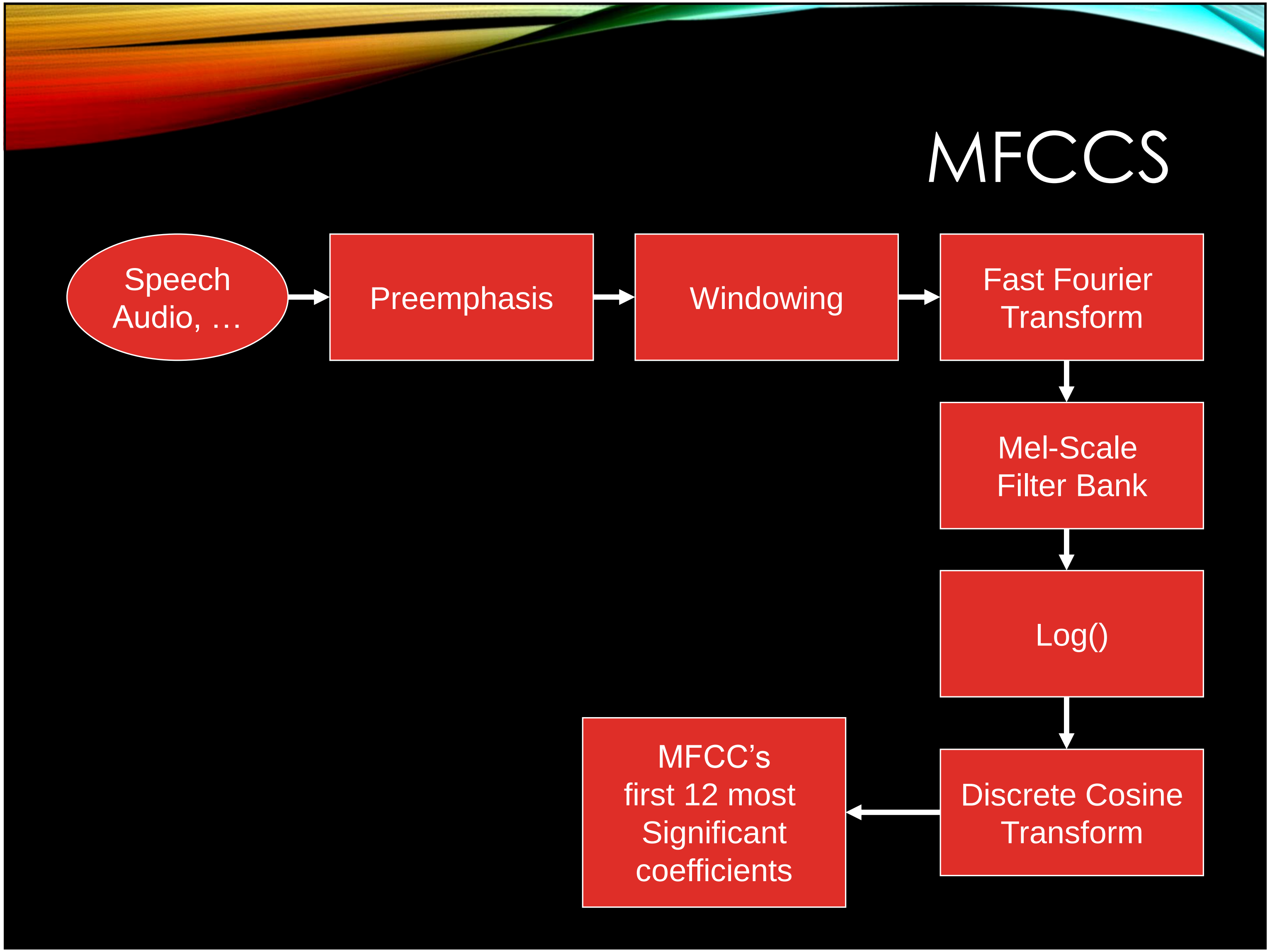
- Mel-scaled filter banks

De-correlating properties

- by applying a discrete cosine transform (which is close to a Karhunen-Loeve transform) a de-correlation of the mel-scale filter log-energies results
- => probabilistic modeling on these de-correlated coefficients will be more effective.

One of the most successful features for speech recognition, speaker recognition, and other speech related recognition tasks.

[1, pp 712-717]


MFCCS
Speech Audio, …
Preemphasis
Windowing
Fast Fourier Transform
Mel-Scale Filter Bank
Log()
Discrete Cosine Transform
MFCC's first 12 most Significant coefficients

MACHINE LEARNING METHODS

- k Nearest Neighbors
- Random Forests (weighted neighborhoods scheme)
- Vector Quantization
  - Finite code book of spectral shapes
  - The code book codes for ‘typical’ spectral shape
  - Method for all spectral representations (e.g. Filter Banks, LPC, ZCR, etc. …)
- Support Vector Machines
- Markov Models
- Hidden Markov Models
- Neural Networks Etc.

VECTOR QUANTIZATION

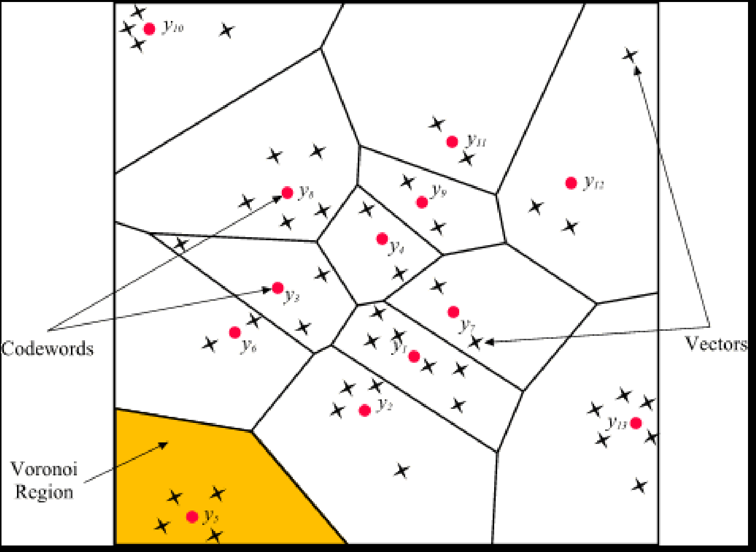
- Data represented as feature vectors.
- Vector Quantization (VQ) Training set to determine a set of code words that constitute a code book.
- Code words are centroids using a similarity or distance measure d.
- Code words together with measure d divide the space into Voronoi regions.
- A query vector falls into a Voronoi region and will be represented by the respective code word.

[2, pp. 466 – 467]

VECTOR QUANTIZATION

Distance measures d(x,y):

- Euclidean distance
- Taxi cab distance
- Hamming distance
- etc.



VECTOR QUANTIZATION

Let a training set of L vectors be given for a certain class of objects.
Assume a codebook of M code words is wanted for this class.

Initialize:

- choose M arbitrary vectors of the L vectors of the training set.
- This is the initial code book.

Nearest Neighbor Search:

- for each training vector v, find the code word w in the current code book that is closest and assign v to the corresponding cell of w.

Centroid Update:

- For each cell with code word w determine the centroid c of the training vectors that are assigned to the cell of w.
- Update the code word w with the new vector c.

Iteration:

- repeat the steps **Nearest Neighbor Search** and **Centroid Update** until the average distance between the new and previous code words falls below a preset threshold.

VECTOR CLASSIFICATION

For an M-vector code book CB with codes

$$CB = \{y_i \mid 1 \le i \le M\},$$

the index m^* of the best codebook entry for a given vector v is:

$$m^* = \underset{1 \le i \le M}{\arg\min}\; d(v, y_i)$$

VQ FOR CLASSIFICATION

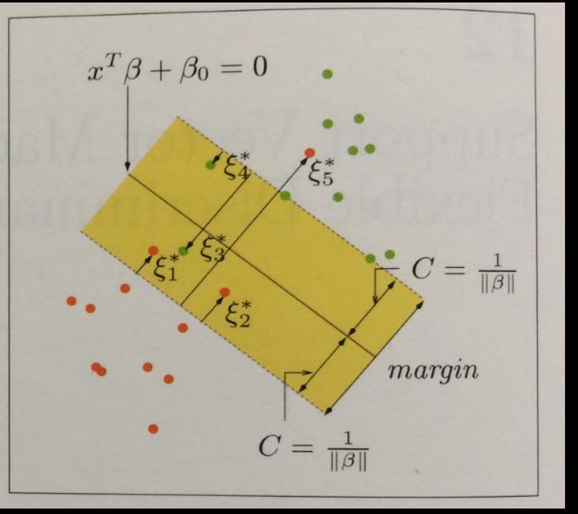
A code book $CB_k = \{y^k_i \mid 1 \leq i \leq M\}$, can be used to define a class C_k.

Example Audio Classification:

- Classes 'crowd', 'car', 'silence', 'scream', 'explosion', etc.
- Determine by using VQ code books CB_k for each of the respective classes C_k.
- VQ is very often used as a baseline method for classification problems.

SUPPORT VECTOR MACHINES

- A generalization of linear decision boundaries for classification.
- Necessary when classes overlap when using linear decision boundaries (non separable classes).



From: [2]

Find hyper plane P: $x^T\beta + \beta_0 = 0$, such that

$\|\beta\|$ is minimized over $\begin{cases} y_i(x_i^T\beta + \beta_0) \geq 1 - \varepsilon_i \ \forall i \\ \varepsilon_i \geq 0, \quad \sum \varepsilon_i \leq constant \end{cases}$

Where $(x_1,y_1), \ldots (x_N,y_N)$ are our training pairs, with $x_i \in \mathbb{R}^p$ and $y_i \in \{-1,1\}$,

$\varepsilon = (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_N)$ are the slack variables, i.e.,

ε_i = the amount that x_i is on the wrong side of the margin $C = \frac{1}{\|\beta\|}$ from the hyper plane P.

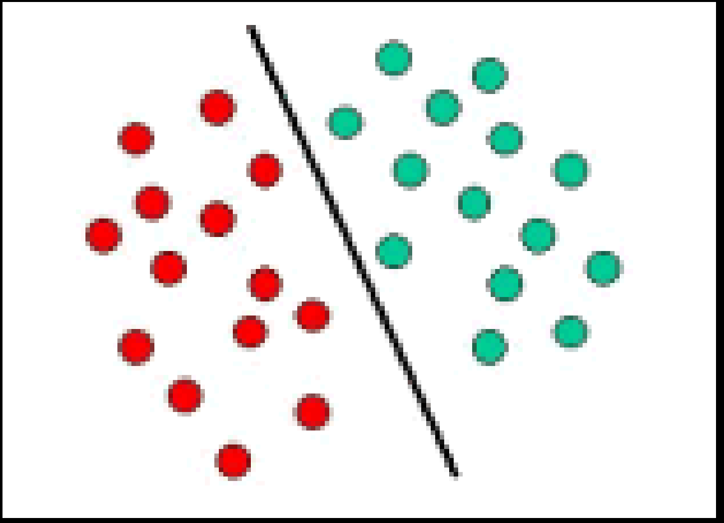
i.e. C is maximized.

=> Problem is quadratic with linear inequalities constraint. [2, pp 377-389]

SUPPORT VECTOR MACHINE (SVM)

In this method so called support vectors define decision boundaries for classification and regression.

An example where a straight line separates the two Classes: a linear classifier



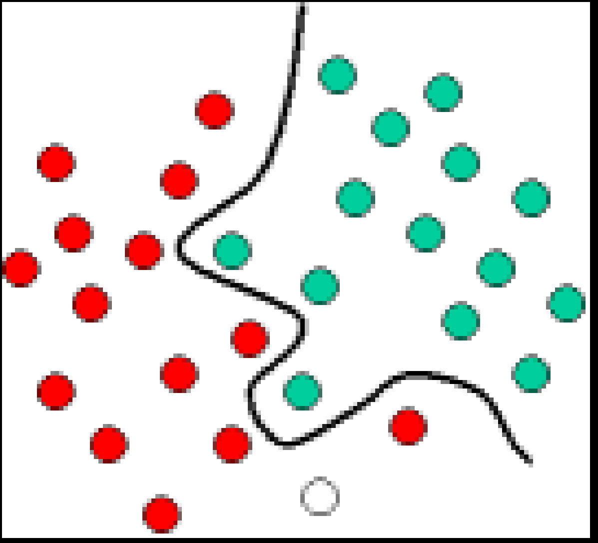
Images from: www.statsoft.com.

SUPPORT VECTOR MACHINE (SVM)

In general classification is not that simple.

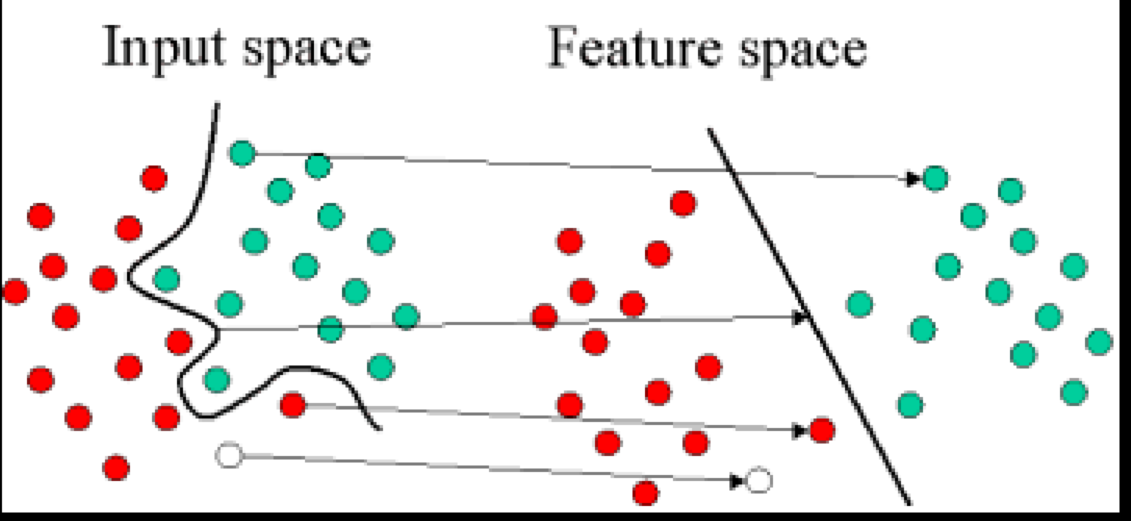
SVM is a method that can handle the more complex cases where the decision boundary requires a curve.

SVM uses a set of mapping functions (kernels) to map the feature space into a transformed space so that hyperplanes can be used for the classification.



SUPPORT VECTOR MACHINE (SVM)

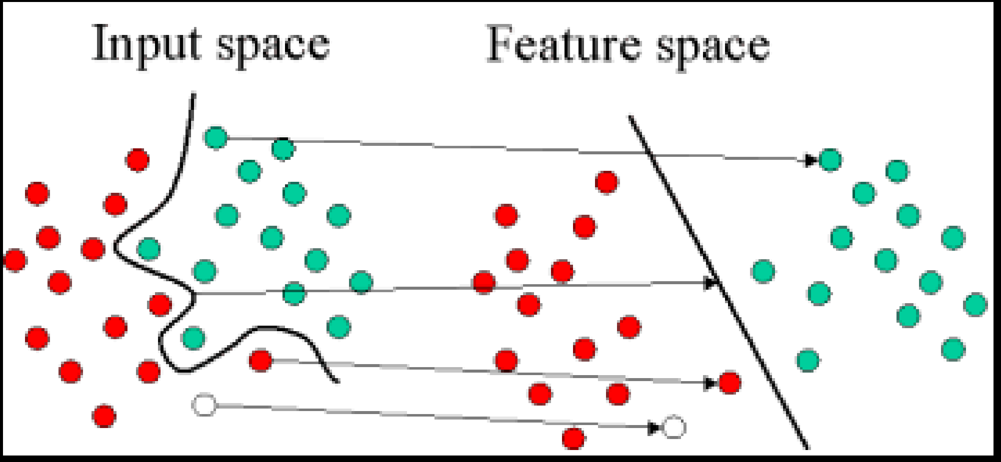
SVM uses a set of mapping functions (kernels) to map the feature space into a transformed space so that hyperplanes can be used for the classification.



SUPPORT VECTOR MACHINE (SVM)

Training of an SVM is an iterative process:

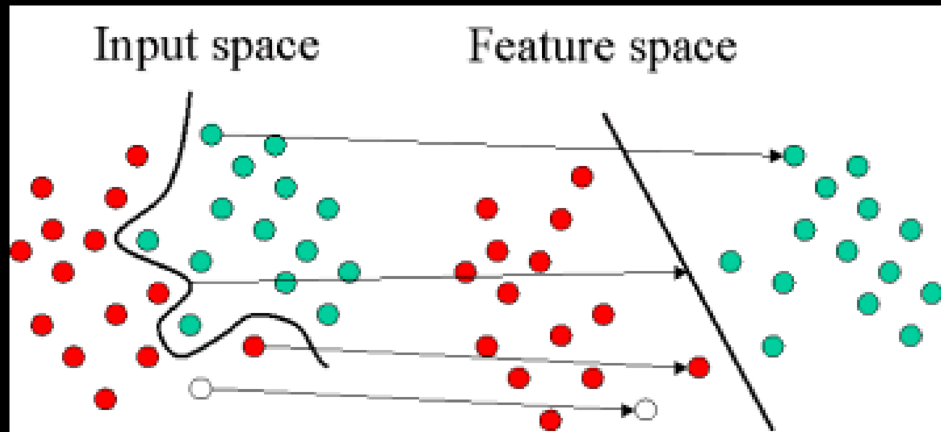
- optimize the mapping function while minimizing an error function
- The error function should capture the penalties for misclassified, i.e., non separable data points.



SUPPORT VECTOR MACHINE (SVM)

SVM uses kernels that define the mapping function used in the method. Kernels can be:

- Linear
- Polynomial
- RBF
- Sigmoid
- Etc.



- RBF (radial basis function) is the most popular kernel, again with different possible base functions.
- The final choice depends on characteristics of the classification task.

AUDIO CLASSIFICATION USING NEURAL NETWORKS

An example by Rishi Sidhu:

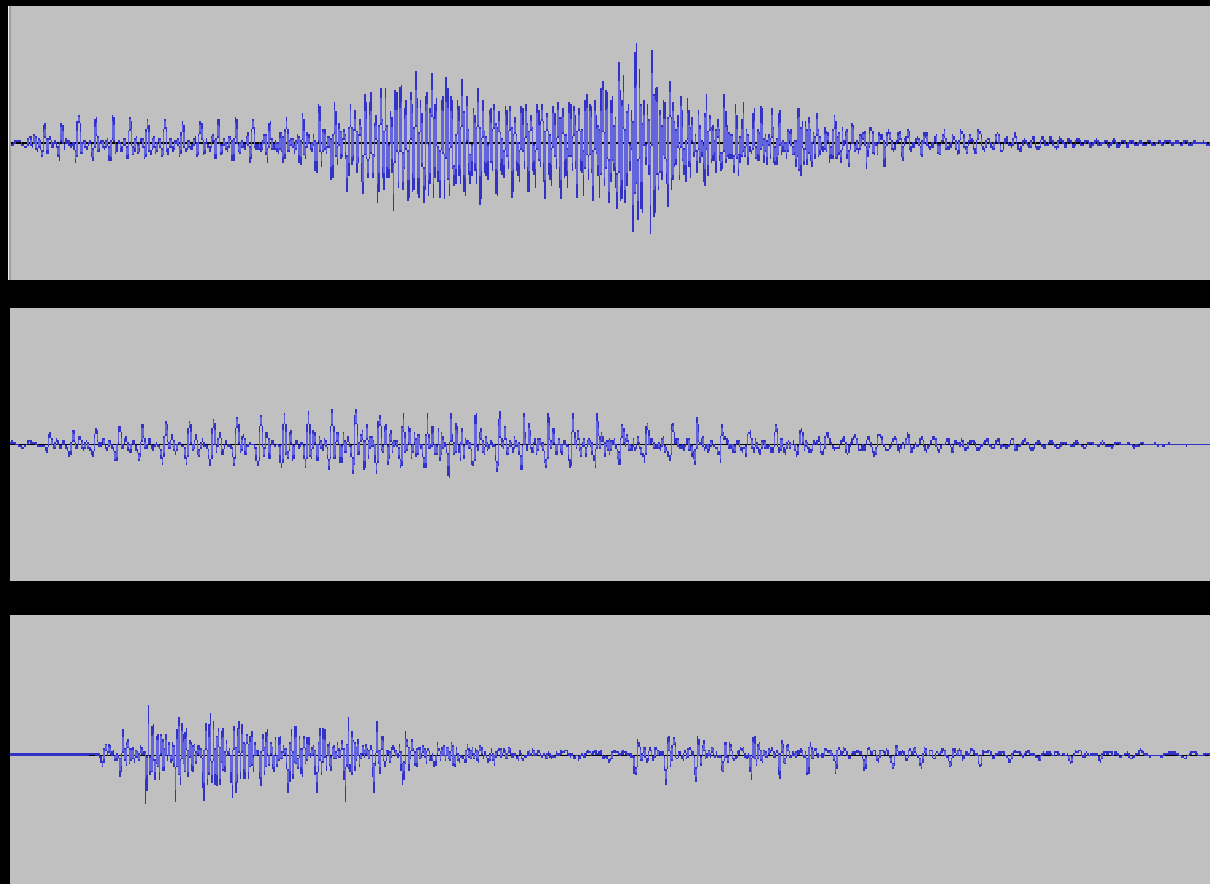
https://medium.com/x8-the-ai-community/audio-classification-using-cnn-coding-example-f9cbd272269e

Using data from the **Spoken Digit Dataset** by Zohar Jackson:

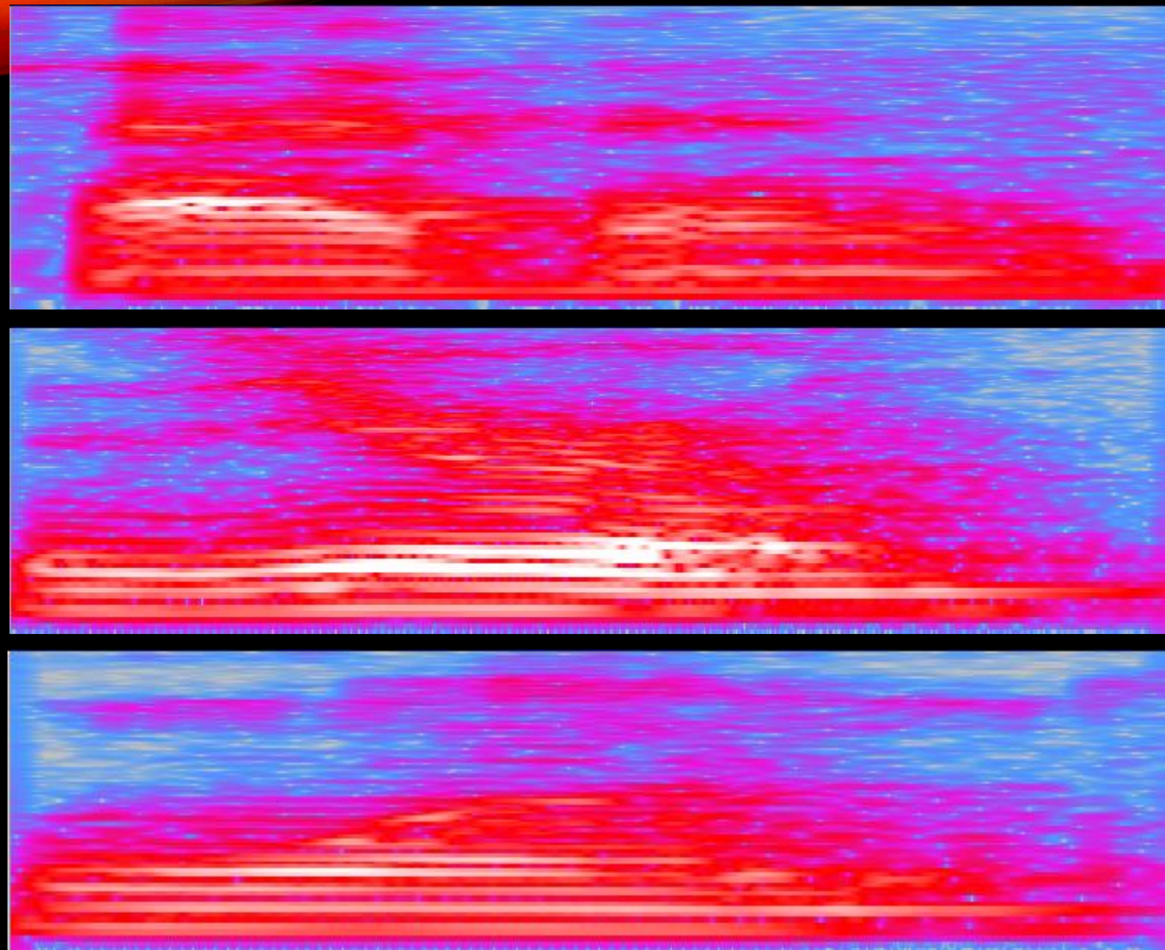
Https://github.com/Jakobovski/free-spoken-digit-dataset

Using Convolutional Neural Networks on Spectograms.

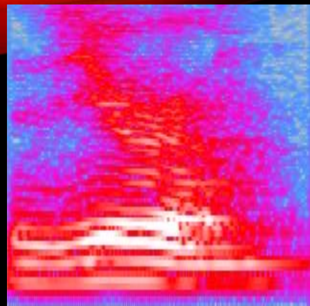
DIGITS

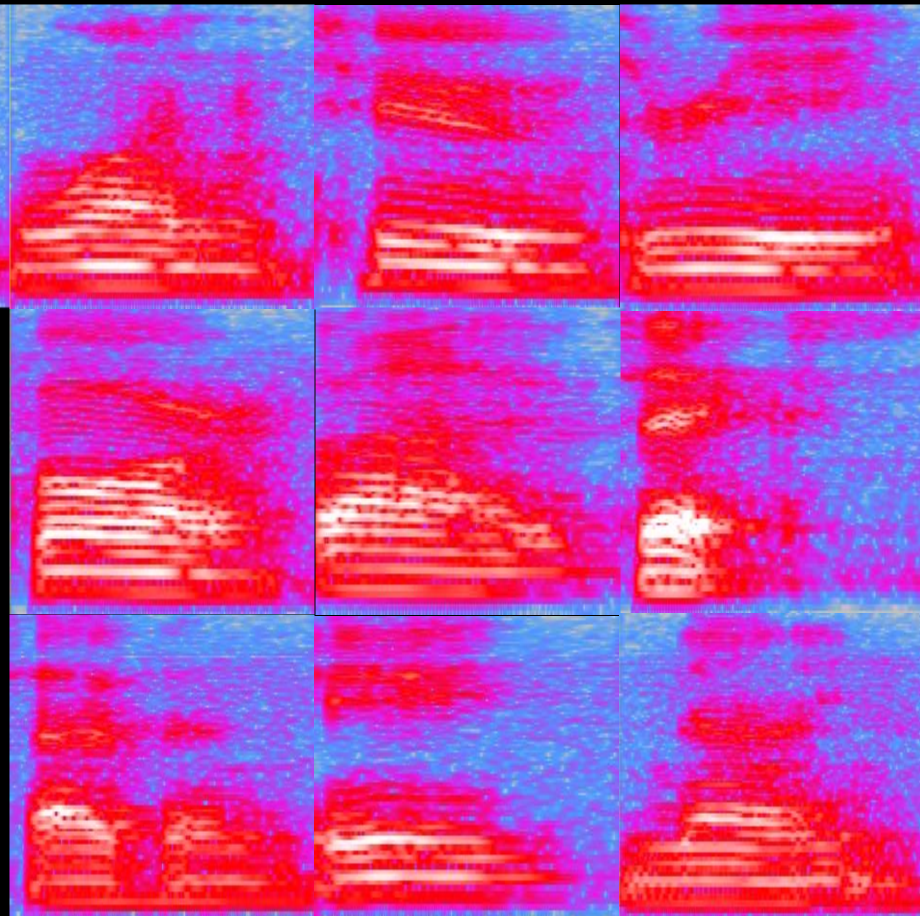


API

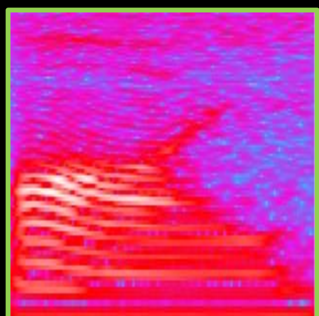


API

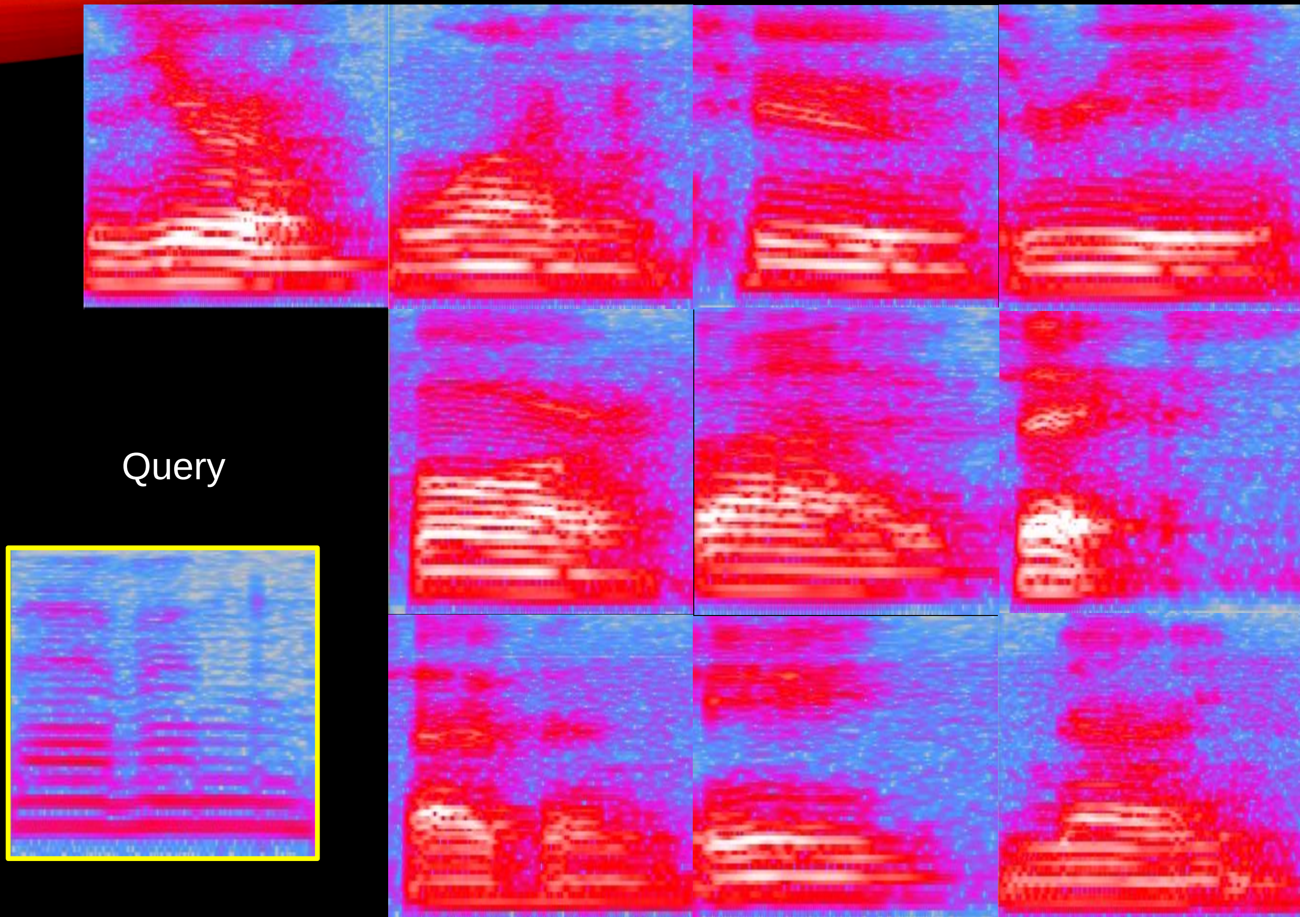




Query



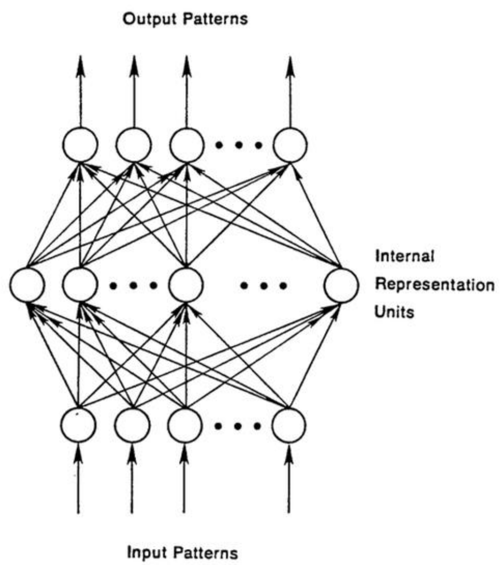
API



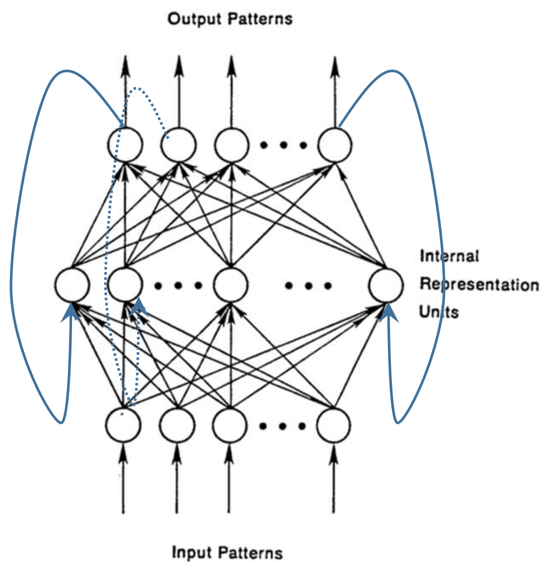
Query

API

Some Neural Networks



Feed Forward Neural Network



Recurrent Neural Network

DNN: AlexNet, VGG16, ResNet, etc.

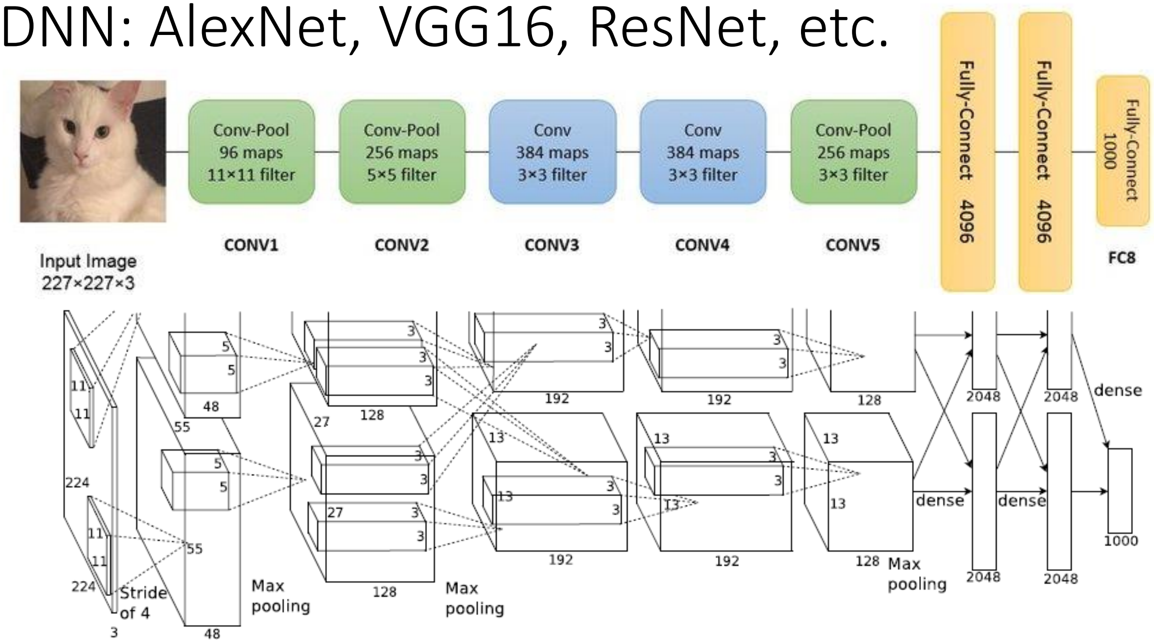


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffrey E. "ImageNet classification with deep convolutional neural networks" Communications of the ACM. 60 (6): 84–90.

Deep Visualization Toolbox

yosinski.com/deepvis

#deepvis

Jason Yosinski

Jeff Clune

Anh Nguyen

Thomas Fuchs

Hod Lipson

Cornell University

University of Wyoming

Jet Propulsion Laboratory
California Institute of Technology

Cats and Dogs

Kaggle Dataset (https://www.kaggle.com/c/dogs-vs-cats/data)

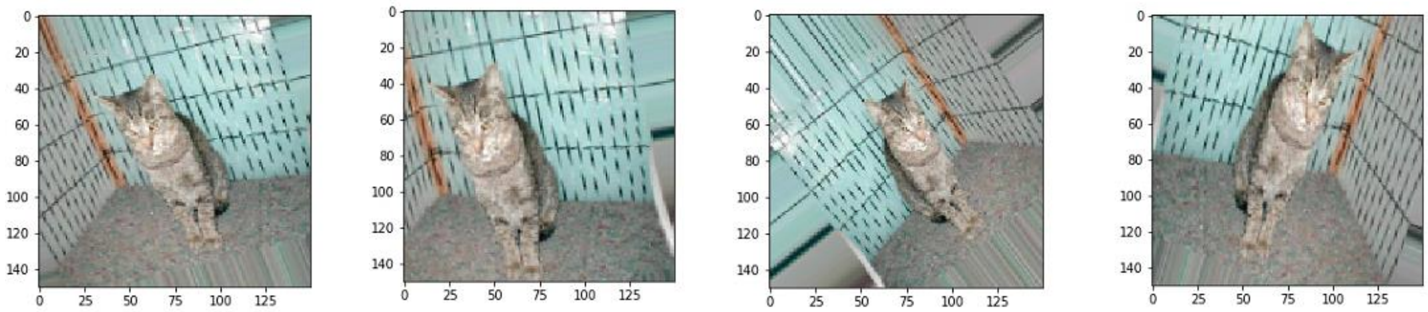
- 2000 images of cats
- 2000 images of dogs

- Given an image: is it a cat or a dog?

Divide into:

- Training set (2000 images)
- Validation set (1000 images)
- Test set (1000 images)

Cats and Dogs



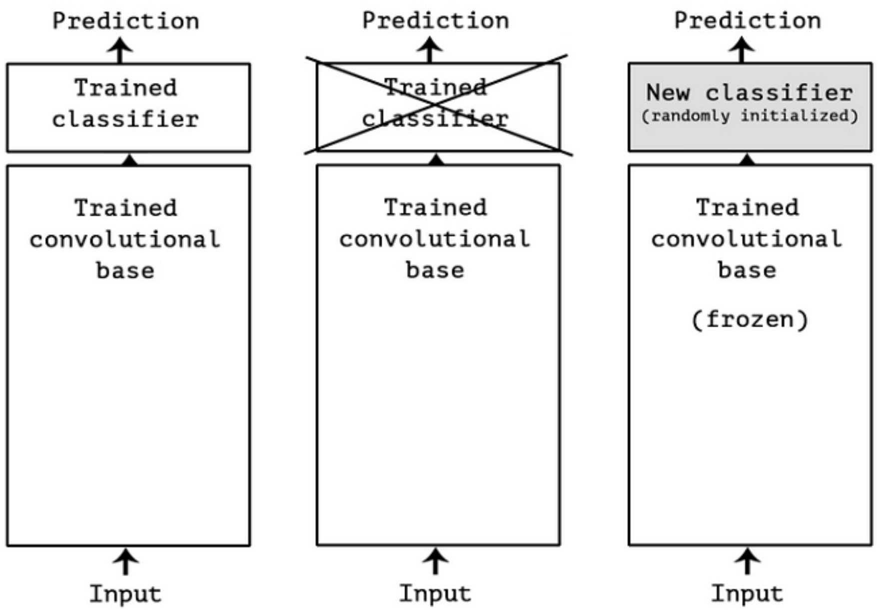
Convolutional Neural Network

- Without any regularization: ~71% accuracy
- With data augmentation: ~82% accuracy
- Feature extraction using a pre-trained NN: ~90% accuracy
- Fine tuning a pre-trained NN: ~95% accuracy

These are examples of Deep Learning with Small Datasets.

Cats and Dogs

VGG16 (pre packed with Keras)

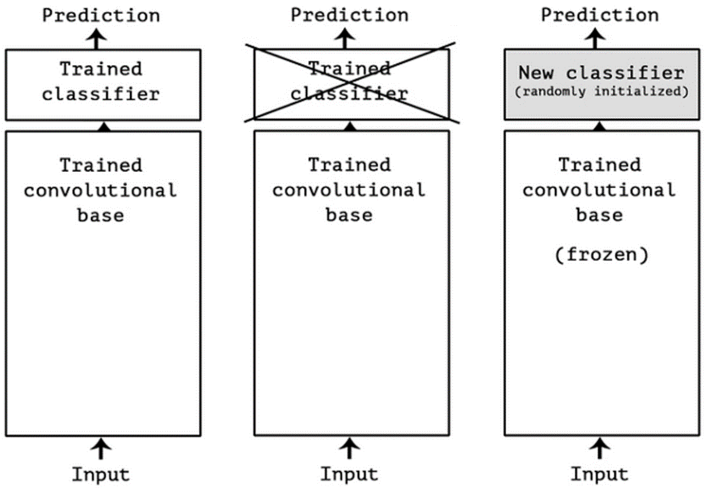


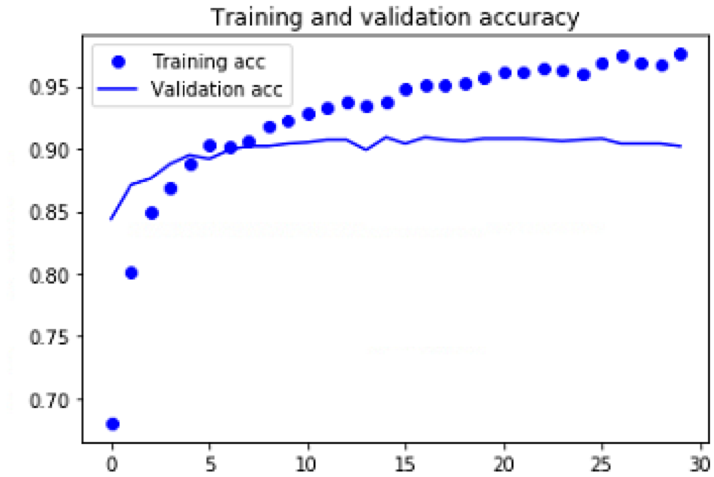
Convolutional Neural Network

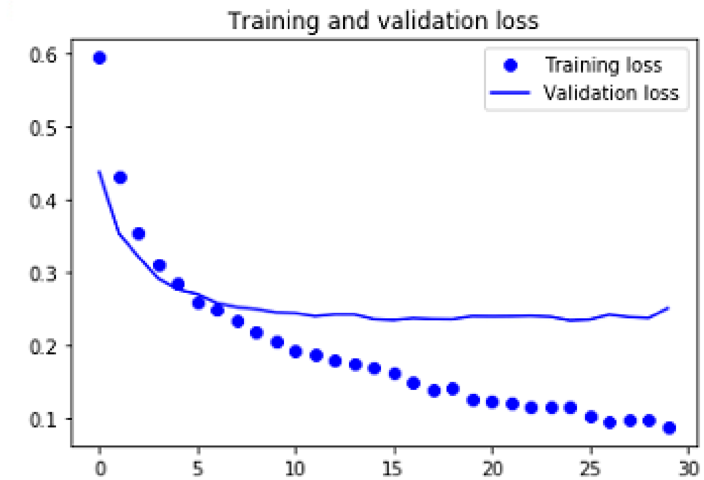
- Without any regularization: ~71% accuracy
- With data augmentation: ~82% accuracy
- Feature extraction using a pre-trained NN: ~90% accuracy
- Fine tuning a pre-trained NN: ~95% accuracy

These are examples of Deep Learning with Small Datasets.

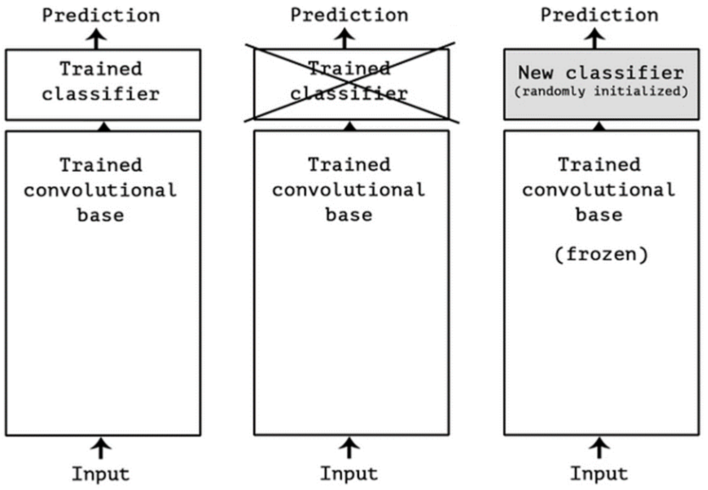
VGG16
Feature Extraction

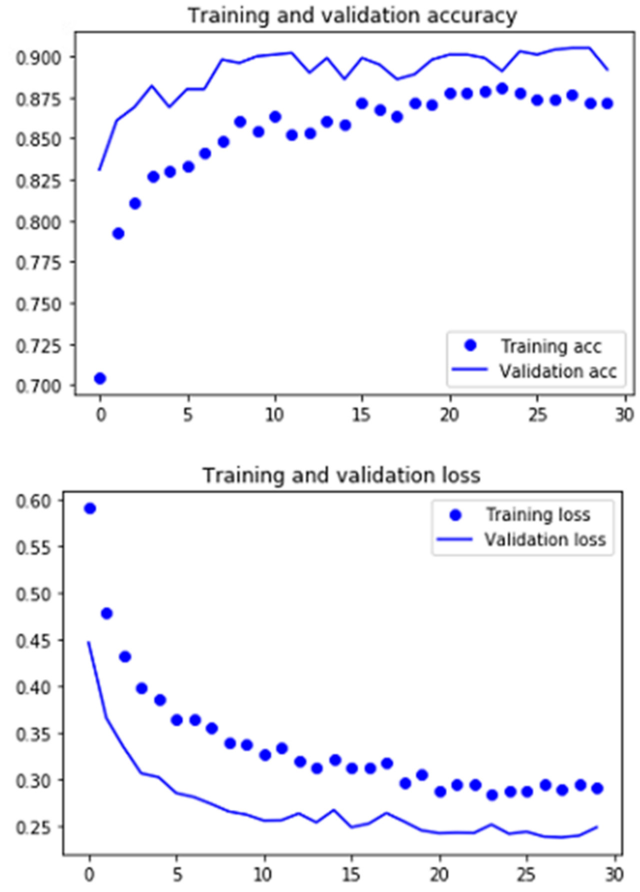






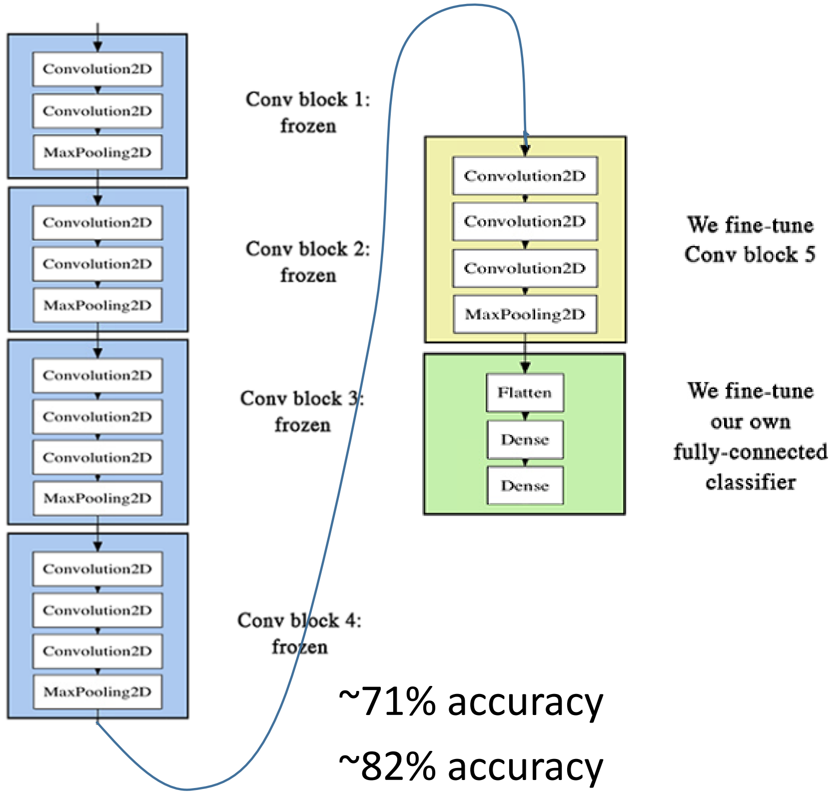
VGG16
Feature Extraction +
Data Augmentation





Cats and Dogs

VGG16 (pre packed with Keras)

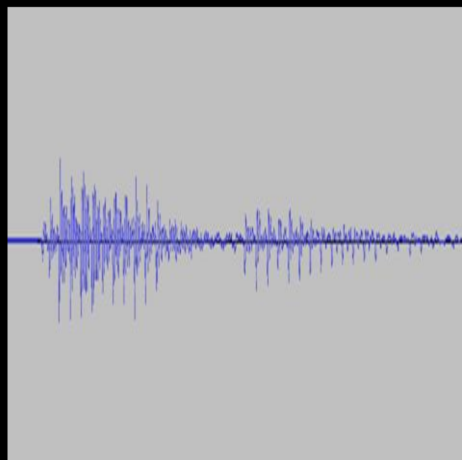


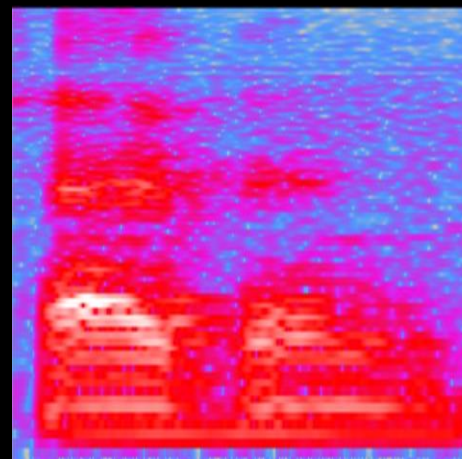
Convolutional Neural Network

- Without any regularization: ~71% accuracy
- With data augmentation: ~82% accuracy
- Feature extraction using a pre-trained NN: ~90% accuracy
- Fine tuning a pre-trained NN: ~95% accuracy

These are examples of Deep Learning with Small Datasets.

CNN'S FOR AUDIO CLASSIFICATION





- Both images can be used to recognize the spoken digit.
- The spectrogram yields better accuracy for the tests.

API

CNN ARCHITECTURE

Input Layer

↓

Convolutional layer with kernel size 3x3

↓

Convolutional layer with kernel size 3x3

↓

Max Pooling layer with pool size 2x2

↓

Dropout layer

↓

Flattening layer

↓

Dense layer 1

↓

Dense layer 1

API

CNN DEFINED IN TF.KERAS

```
#Define Model
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
#Compile
model.compile(loss=keras.losses.categorical_crossentropy,
optimizer=keras.optimizers.adam(), metrics=['accuracy'])
print(model.summary())
#Train and Test The Model
model.fit(x_train, y_train, batch_size=4, epochs=10, verbose=1, validation_data=(x_test,
y_test))
```

API

TRAINING, TEST AND VALIDATION DATASETS

Training Data

- 1800 Images of Spectrograms: 34x50 pixels
- Each image is labeled with the correct digit

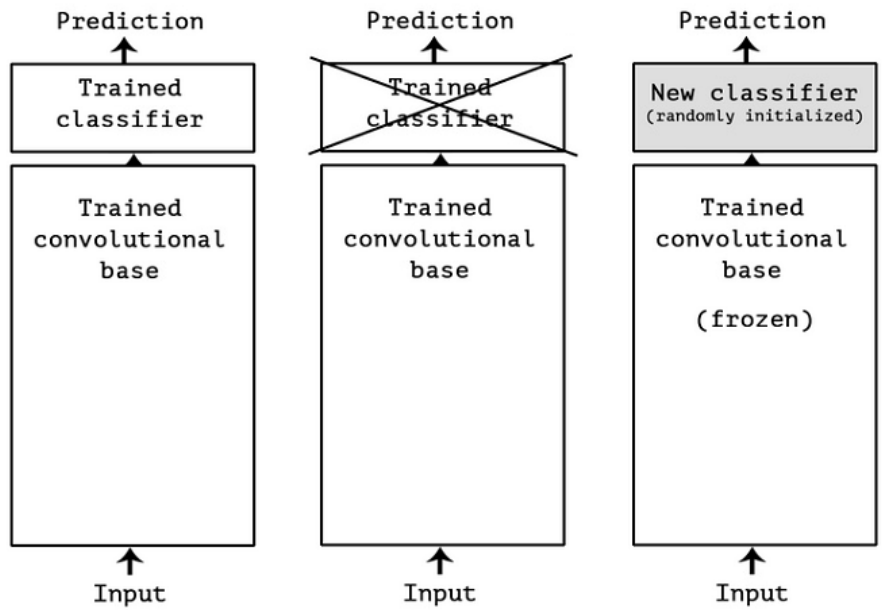
Validation Data

- 200 Images of Spectrograms: 34x50 pixels
- Each image is labeled with the correct digit
- Exclusive speaker(s)

Test Data

- 200 Images of Spectrograms: 34x50 pixels
- Each image is labeled with the correct digit
- Exclusive speaker(s)

Digits



VGG16 (pre packed with Keras)

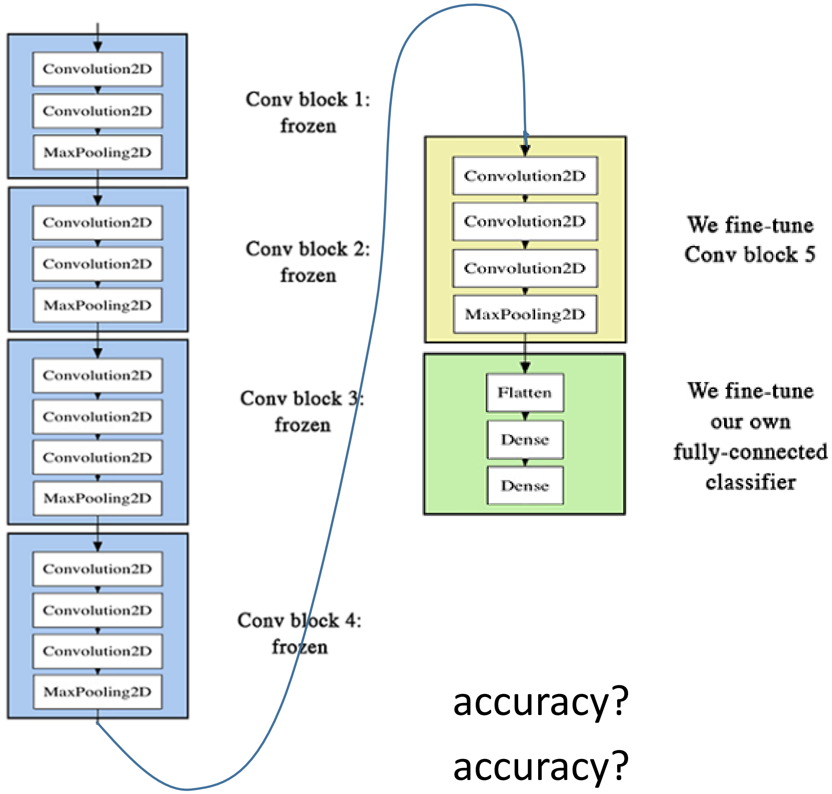
Convolutional Neural Network

- Without any regularization: accuracy ?
- With data augmentation: accuracy ?
- Feature extraction using a pre-trained NN: accuracy ?
- Fine tuning a pre-trained NN: accuracy ?

These are examples of Deep Learning with Small Datasets.

Digits

VGG16 (pre packed with Keras)



Convolutional Neural Network

- Without any regularization: accuracy?
- With data augmentation: accuracy?
- Feature extraction using a pre-trained NN: accuracy?
- Fine tuning a pre-trained NN: accuracy?

These are examples of Deep Learning with Small Datasets.

W. Chunyang et al. Transformer-based Acoustic Modeling for Streaming Speech Synthesis, INTERSPEECH 2021

https://transformer-tts-accoustic-model.github.io/samples/

Tacotron2 uses Bi-directional Long Short-term Memory (BLSTM) recurrent networks.

- cannot effectively model long-term dependencies
- a poor quality on long speech.

FastSpeech state-of-the-art

- in modeling speech prosody and spectral features, but
- computation is parallel over the full utterance context.

W. Chunyang et al. Transformer-based Acoustic Modeling for Streaming Speech Synthesis, INTERSPEECH 2021

TTS systems usually consist of two stages:

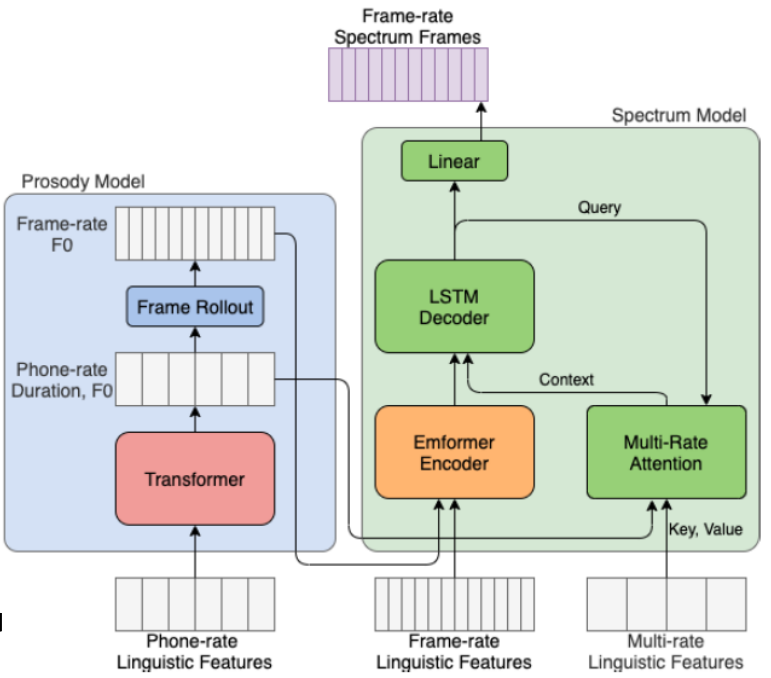
- acoustic model that predicts the prosody and spectral features
- followed by a neural vocoder that generates the audio
- waveform.

Tranformer models:

- model long-term dependencies
- Complexity grows quadratically

This work

- Effcient constant speed implementation: for streaming speech synthesis
- uses a transformer network that predicts the prosody features at phone rate
- an Emformer network to predict the frame-rate spectral features (streaming)
- WaveRNN Vocoder used



https://transformer-tts-accoustic-model.github.io/samples/

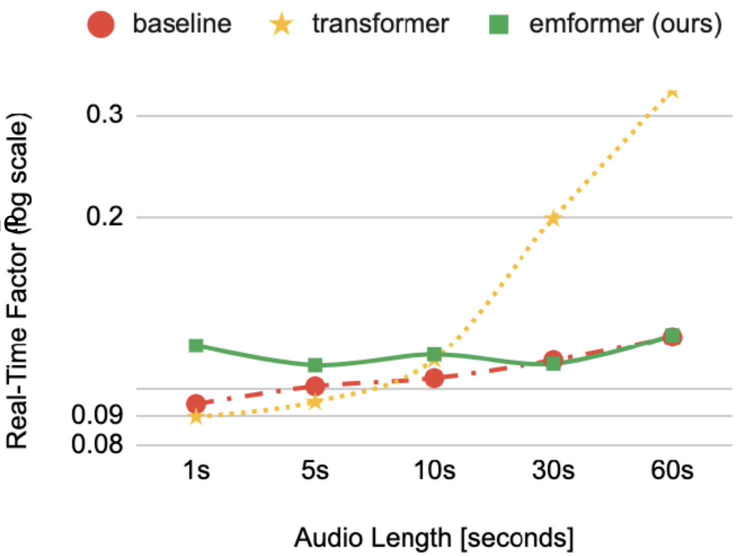
W. Chunyang et al. Transformer-based Acoustic Modeling for Streaming Speech Synthesis, INTERSPEECH 2021

TTS systems usually consist of two stages:

- acoustic model that predicts the prosody and spectral features
- followed by a neural vocoder that generates the audio
- waveform.

Tranformer models:

- model long-term dependencies
- Complexity grows quadratically



System	Prosody	Spectrum	Normal	Long
Groundtruth	–	–	4.307 ± 0.037	4.360 ± 0.044
Baseline [11]	BLSTM with self-attention [26]	Multi-rate attention [11]	4.173 ± 0.042	4.019 ± 0.055
Ours-1	Transformer	Multi-rate attention	4.174 ± 0.042	4.107 ± 0.052
Ours-2	BLSTM with self-attention	Emformer with multi-rate attention	4.192 ± 0.041	4.034 ± 0.053
Ours-3 (best)	Transformer	Emformer with multi-rate attention	**4.213 ± 0.042**	**4.201 ± 0.048**

https://transformer-tts-accoustic-model.github.io/samples/

REFERENCES

1. T.F. Quatieri, Discrete-Time Speech Signal Processing, Principles and Practice, Prentice-Hall, Inc. 2002.
2. T. Hastc, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Data Mining, Inference, and Prediction, Springer, 2001.
3. W.H. Press, S.A.Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipies in C++, The Art of Scientific Computing, 2nd Edition, Cambridge University Press, 2002.
4. S.B. Davies, P. Mermelstein, Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences, IEEE Trans. Acoustics, Speech, and Signal Processing, vol. ASSP-28, no.4, pp. 357-366, Aug. 1980.

REFERENCES

5. P. Kenny, "Joint Factor Analysis of Speaker and Session Variability: Theory and Algorithms, Tech. Report CRIM-06/08-13," 2005.

Available: http://www.crim.ca/perso/patrick.kenny

6. N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Frontend factor analysis for speaker verification," IEEE Trans. Audio, Speech, Lang. Process., vol. 19, no. 4, pp. 788–798, May 2011.

7. François Chollet, Deep Learning with Python, Manning Publications, November 2017.