AGH University of Science and Technology

Deep Learning in Speech Technology

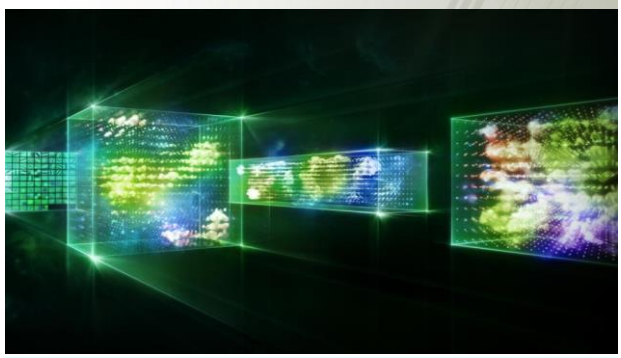
Lectures

Jakub Gałka, Department of Electronics ©
jgalka@agh.edu.pl

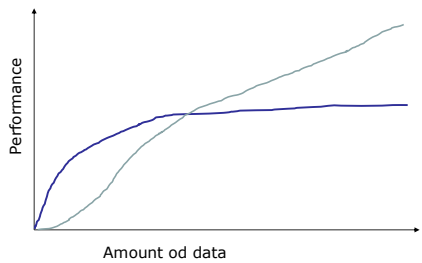
Outline

1. What is Deep Learning
2. Deep Learning tools
3. Deep Network Architectures and Applications
   - Feed-Forward Networks (FF)
   - Recurrent Neural Networks (RNN)
   - Long Short-Term Memory Networks (LSTM)
   - Convolutional Neural Networks (CNN)
   - Residual Neural Networks
   - Autoencoders
4. Learning architectures and approaches
   - Classification, discrimination, regression
   - Transfer Learning
   - Generative-Adversarial Networks (GAN)

Why going deep ?



Why Deep Learning ?



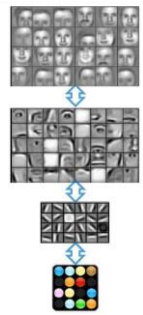
Why Now ? (2010+)

- More data
- More computational power
- More interest, more people
- Better algorithms
- Better results
- More applications

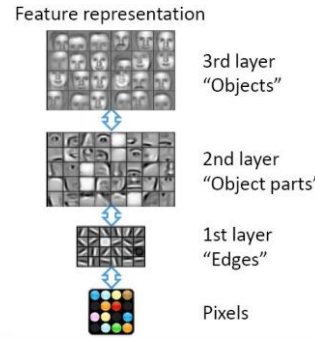
What is Deep Learning

WiKi:

- Deep learning is a class of machine learning algorithms
- use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input.
- learn in supervised (e.g., classification) and/or unsupervised (e.g., pattern analysis) manners.
- learn multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts.



Layered abstraction

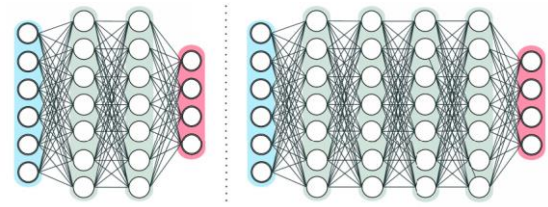


How To „Deep Learning" ?

Caffe, Caffe2, Chainer, CNTK, MXNet, PaddlePaddle, PyTorch, TensorFlow, theano, torch, WOLFRAM

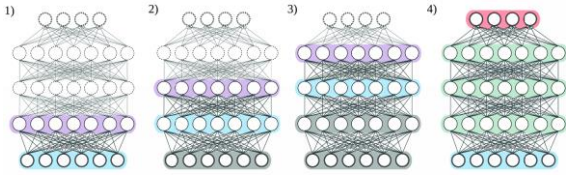
Normal vs Deep Feed-Forward Network (fully connected)

- Input hidden ... hidden output



Deep network training

- Network pretraining (1-3) and Fine-tunning (4)
- Unsupervised and supervised training (loss function)

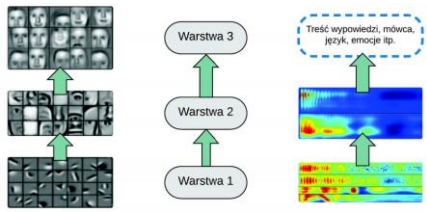


- Regularization (L2, L1, Dropout)
- Data augumentation
- Meta-parameters optimization (Batch size, net size)
- Evaluation (Metric)

```
from keras.models import Sequential
from keras.layers import Dense
import numpy
# fix random seed for reproducibility
numpy.random.seed(7)

dataset = numpy.loadtxt("my_dataset.csv", delimiter=",")
# input (X) and output (Y)
X = dataset[:,0:39]
Y = dataset[:,39]

# create model
# 12, 8, 4, and 1 neuron in consecutive layers
model = Sequential()
# input layer
model.add(Dense(12, input_dim=39, activation='relu'))
model.add(layers.Dropout(0.2, noise_shape=None, seed=None))
model.add(Dense(8, activation='relu'))
model.add(layers.Dropout(0.1, noise_shape=None, seed=None))
model.add(Dense(4, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
# Output layer have one leuron - '1' for Speech, '0' for non-speech frame

# Compile model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Fit the model
model.fit(X, Y, epochs=100, batch_size=20)

# evaluate the model
scores = model.evaluate(X, Y)
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))

# predict output
predictions = model.predict(X)
```

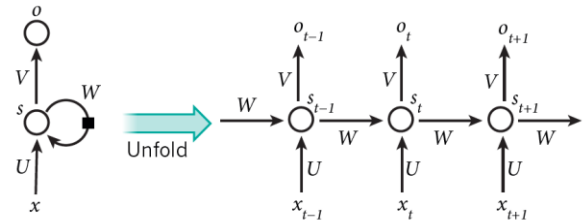
Typical Deep FFN architectures for speech processing

- Input: MFCC, Mel-filterbank, FFT
- Frame stacking at input (broader context)
- Outputs: softmax (prob), classification decision
- Feature extractor
- Problem: No direct support for time-wise/process learning



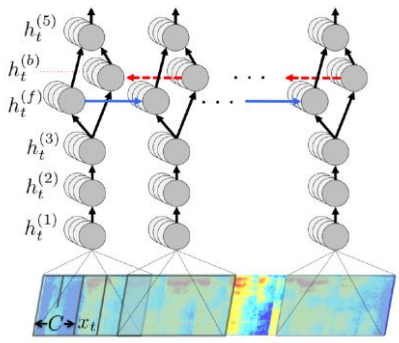
Recurrent Neural Networks

- Ability to remember information



Recurrent Neural Networks

BAIDU (12.2014) Deep Speech: Scaling up end-to-end ASR



Softmax output

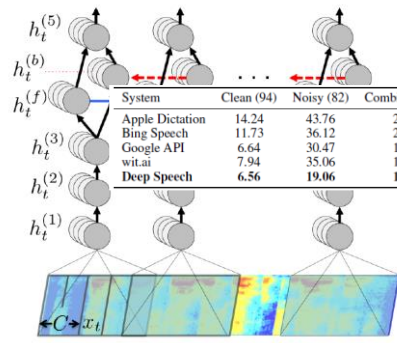
Recurrent
Recurrent

3 clipped-ReLU FF

FFT spectrogram input (5000 hrs)

Recurrent Neural Networks

BAIDU (12.2014) Deep Speech: Scaling up end-to-end ASR



System	Clean (94)	Noisy (82)	Combined (176)
Apple Dictation	14.24	43.76	26.73
Bing Speech	11.73	36.12	22.05
Google API	6.64	30.47	16.72
wit.ai	7.94	35.06	19.41
Deep Speech	**6.56**	**19.06**	**11.85**

Softmax output

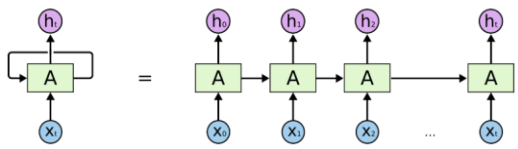
urrent

ipped ReLU FF

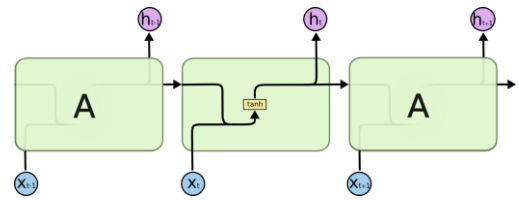
FFT spectrogram input (5000 hrs)

RNN vs LSTM

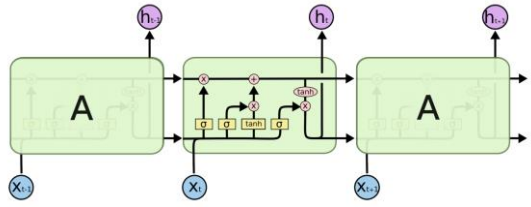
- Vanishing gradient problem
- RNN acts as a very deep FF network



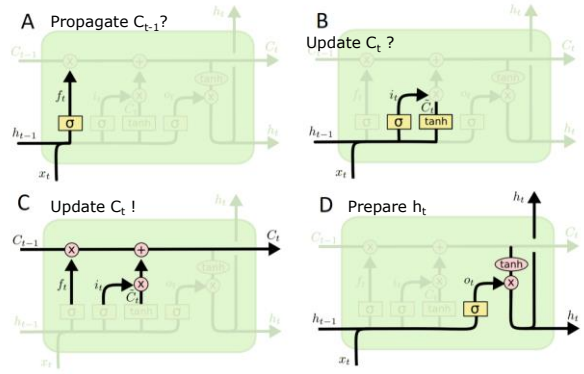
Standard RNN unwrapped



Long Short-Term Memory Networks (1997)

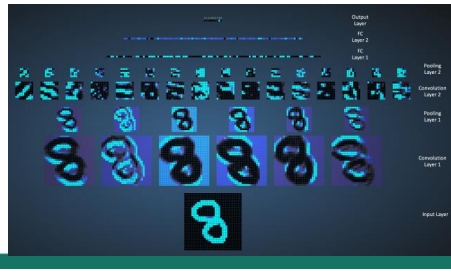


Steps of LSTM operation



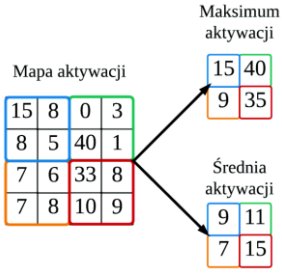
Convolutional Neural Networks

- In fully conected FF networks number of weights is to big for huge inputs.
- Same features can be observed in different places of the same signal (as in images)
- We can **group neurons** to analyse different parts of data –> lower number of weights

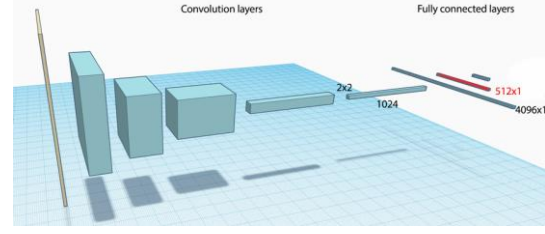


CNN workflow

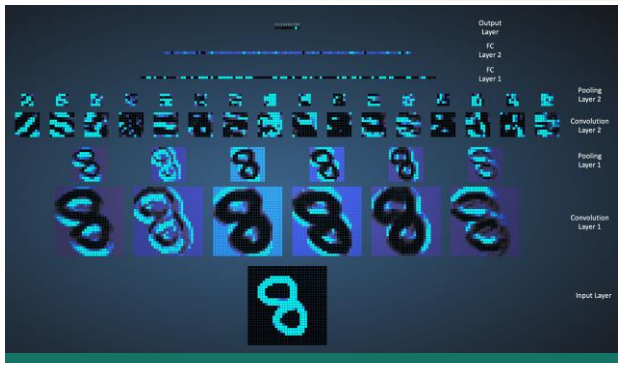
Max-pooling, Average-pooling



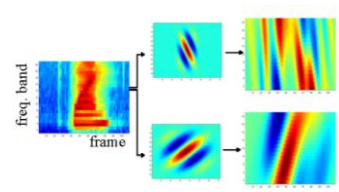
Exemple of typical CNN-FC model



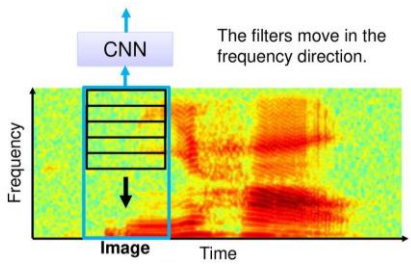
CNN classification



CNN audio classification



CNN audio event classification example



CNN concepts

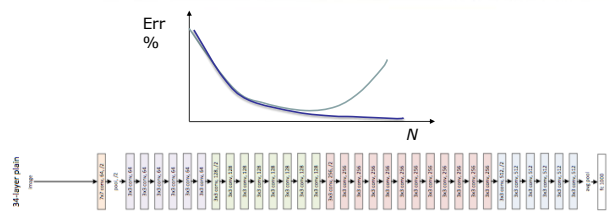
- Kernel / filter size
- Number of kernels
- Stride – kernel shift (usually 1)
- Padding
- Pooling – reduces dimensionality
  Max-pooling, Average-pooling
- Flattening

CNN in Keras

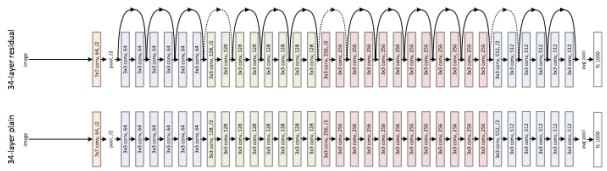
```
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

```
# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (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'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

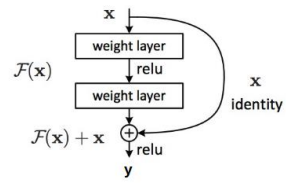
ResNet - Residual Networks



ResNet - Residual Networks



ResNet - Residual Networks

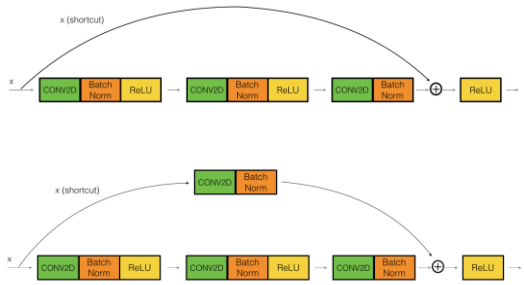


$$y = x + F(x)$$

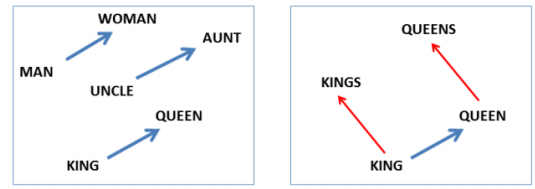
$$\frac{\delta E}{\delta x} = \frac{\delta E}{\delta y} * \frac{\delta y}{\delta x} = \frac{\delta E}{\delta y} * (1 + F'(x))$$

$$= \frac{\delta E}{\delta y} + \frac{\delta E}{\delta y} * F'(x)$$

```
X_shortcut = X     # Store the initial value of X in a variable
## Here perform convolution + batch norm operations on X
X = Add()([X, X_shortcut])   # SKIP Connection
```

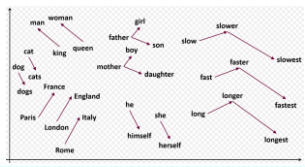


Word embeddings - latent space



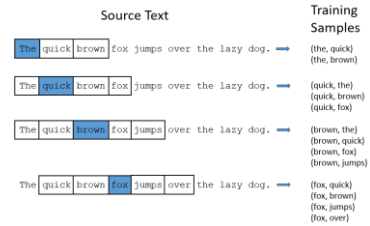
Word2Vec – word embeddings
(Google, Mikolov)

- Converting text into high-dimensional vector of numbers (latent space embedding)
- Preserves linguistic and pragmatic information
- Embeddings are easy to manipulate and use in ML algorithms



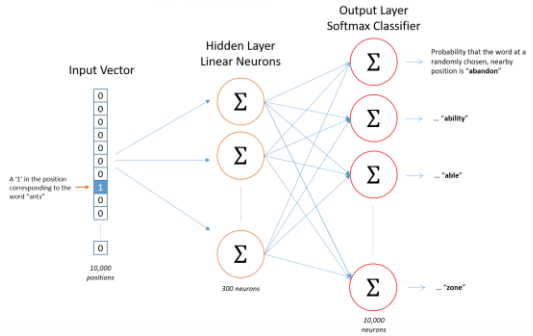
Word2vec algorithm

- Train your network using hot-word representation
- Use skip-gram method, or continuous bag of words



- Use bottleneck feature as embeddings

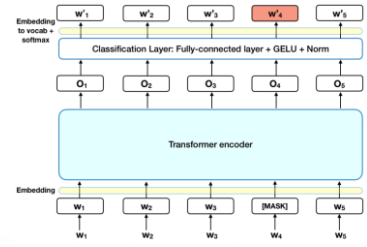
One-hot as NN input



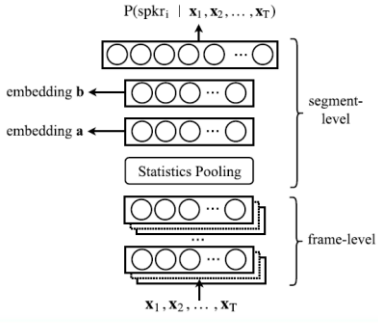
BERT Language Model (Google)

Bidirectional Encoder Representations from Transformers

- Transformer is an attention mechanism that learns contextual relations between words

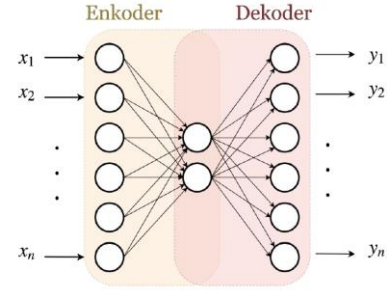


Deep Neural Network Embeddings for Text-Independent Speaker Verification

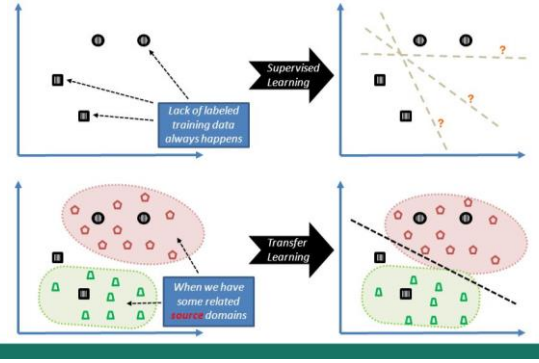


- Soft-max output layer
- Segment T aggregation
- Max +/-8 frame-context
- 2 TDFF ReLU levels
- 3 time-delayed FF ReLU levels
- MFCC input

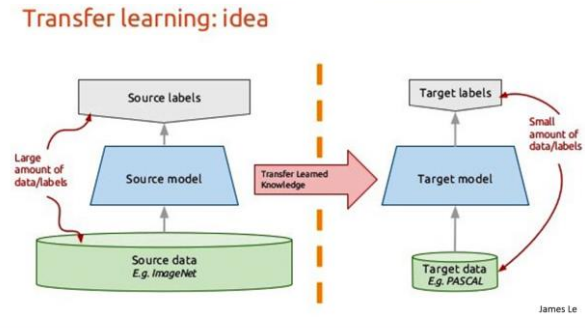
Autoencoders



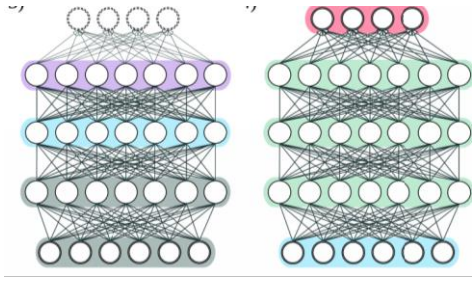
Transfer Learning



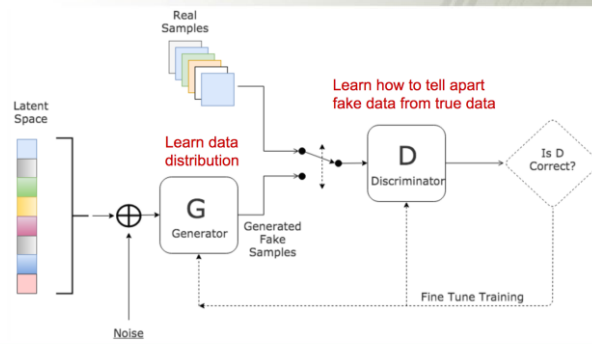
Transfer Learning



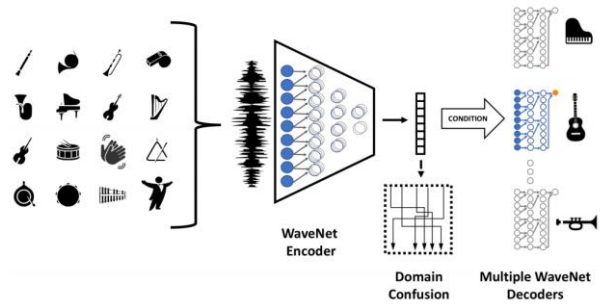
Transfer Learning
possible training approaches



Generative Adversarial Networks



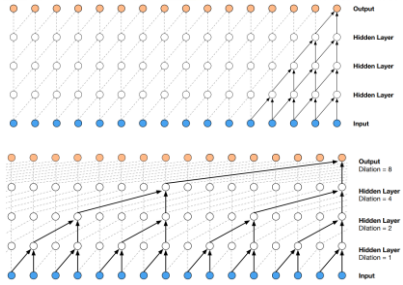
GAN Example from FB
A Universal Music Translation Network
https://www.youtube.com/watch?v=vdxCqNWTpUs&feature=youtu.be



Speech Synthesis/Conversion using Deep Learning

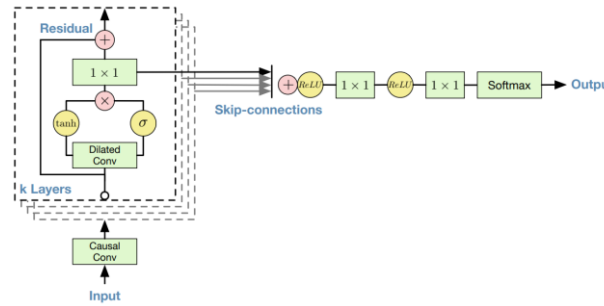
- WaveNet baseline
- Distilled Wavenet, ClariNet (Baidu)
- Tacotron2 (NATURAL TTS SYNTHESIS BY CONDITIONING WAVENET ON MEL SPECTROGRAM PREDICTIONS)
- Lot of implementations (NVIDIA), and downloadable models (eg. Wavenet generator)

Wavenet – sample by sample convolution



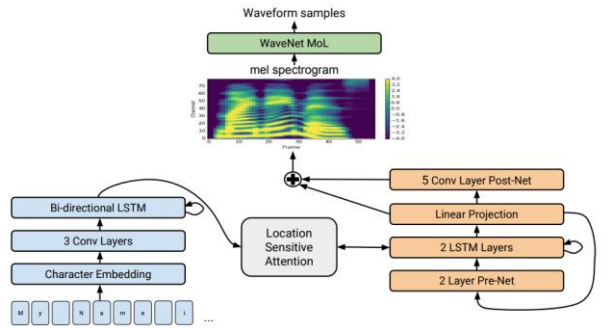
- „At training time, the conditional predictions for all timesteps can be made in parallel because all timesteps of ground truth x are known. When generating with the model, the predictions are sequential: after each sample is predicted, it is fed back into the network to predict the next sample."

Wavenet training architecture



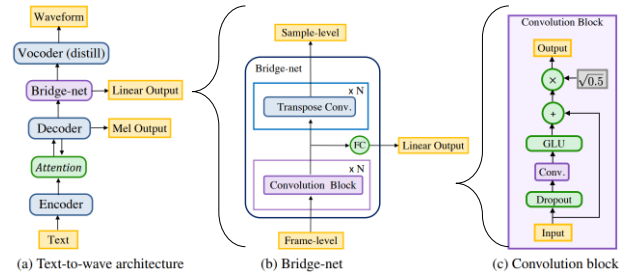
Tacotron 2 – Google TTS

https://google.github.io/tacotron/publications/tacotron2/index.html



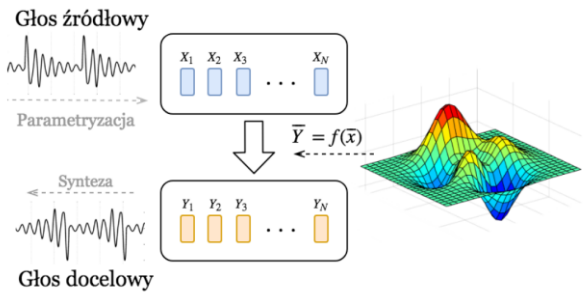
ClariNet: Parallel Wave Generation in End-to-End Text-to-Speech, Baidu

https://clarinet-demo.github.io/



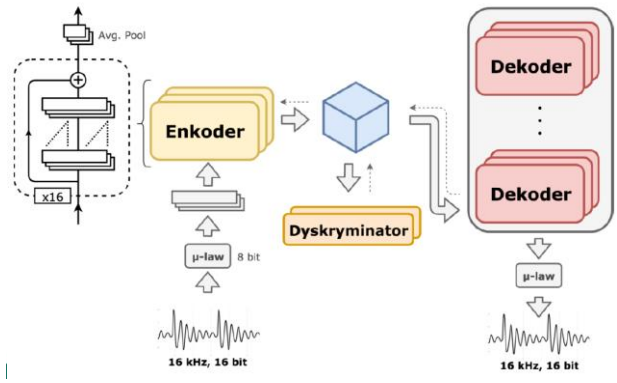
(a) Text-to-wave architecture (b) Bridge-net (c) Convolution block

Voice Conversion



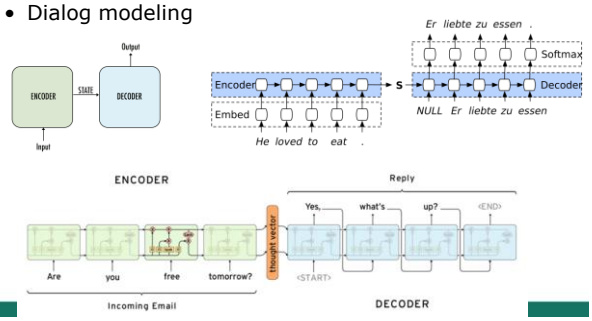
Voice Conversion with GAN

Hubert Siuzdak (IA, 2019)



Sequence-to-sequence modeling

- Used in machine translation
- Dialog modeling



LSTM Sequence-to-sequence translation example