1D Convolutional Neural Networks summary

E. Charalampakis, Prof. Ioannis Pitas
Aristotle University of Thessaloniki
pitas@csd.auth.gr
www.aiia.csd.auth.gr
Version 1.2.1

Artificial Intelligence & Information Analysis Lab

1D Convolutional Neural Networks

- Deep Neural Networks
- 1D Convolution
- 1D CNN Architecture
- Convolutional Layer
- Fully Connected Layer
- Pooling Layers
- Activation Functions
- Supervised Learning
- Classification/Regression
- CNN Training
- 1D CNN applications
  - ECG monitoring
  - Music tagging

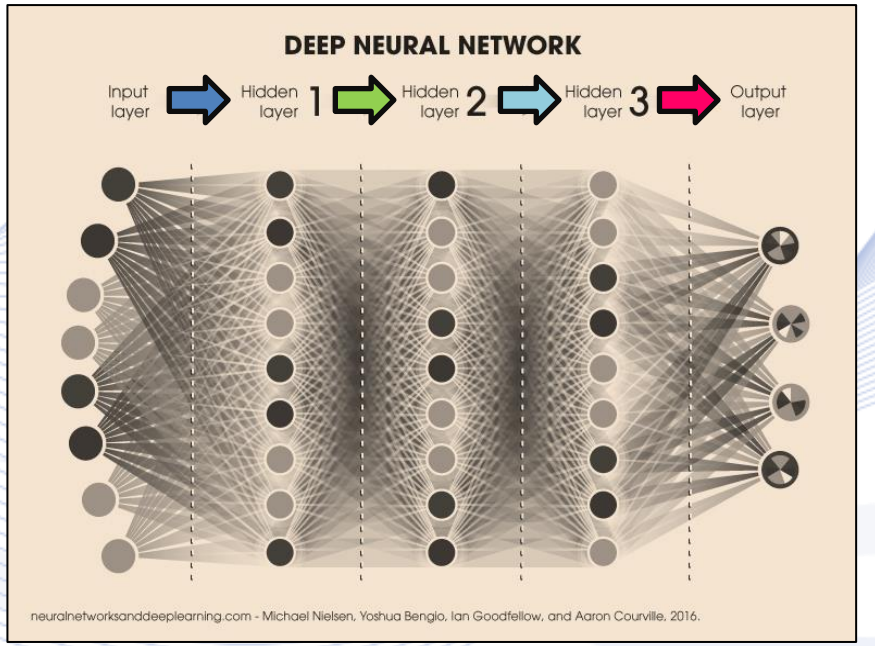
Introduction

During the last decade, Convolutional Neural Networks (CNNs) have become the standard for multiple Computer Vision and Machine Learning operations. Yet, this may not be a viable option in numerous applications over 1D signals especially when the training data is scarce or application specific. 1D CNNs have recently been proposed and immediately achieved great results in several applications.

Deep Neural Networks

Deep Neural Networks (***DNN***) have a count of layers (depth) $L \geq 3$.

- Typically, these networks contain a big number of hidden layers ($L \gg 3$).



Deep Neural Network with $L = 4$

Deep Neural Networks

- The classic neurons have a weighted sum on vectors (1D) of features values:

$$z = w_0 + \sum_{i=1}^{n} w_i x_i = w_0 + \mathbf{w}^T \mathbf{x}.$$

- If for example we input $N_1 \times N_2$ pixel color images are represented by huge dimensionality vectors $n = N_1 \times N_2 \times 3$, they cannot be processed by fully connected Multilayer Perceptrons.
- Problem: Increased computational complexity.
- Solution: Convolutional Neural Networks (CNNs) that employ sparse connectivity and weight replication.

Convolutional Neural Networks

- CNNs produce great results by utilizing 2D convolutions on 2D or 3D input.

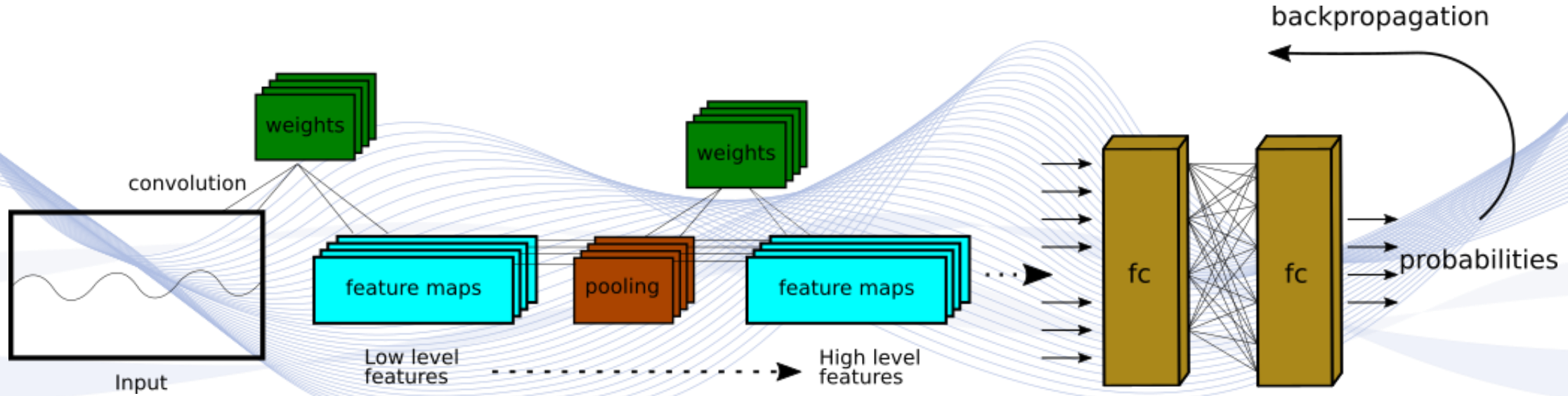
- In various studies [1] it was shown that for dealing with the processing of certain 1D signals, CNNs with 1 dimensional convolution kernels (**1D CNN**) are preferrable to their 2D counterpart.

- Standard 1D CNN input is a 1D signal represented by a ***1D vector X*** that contains samples from an analog signal (e.g., Sound), time series (e.g., Text, Financial data) etc.

Convolutional Neural Networks

1D Convolutional Neural Networks (***1D CNN***):

- Employ 1-dimensional linear convolutions in the first layers, followed by a pooling operation.
- They may employ fully connected MLPs in the last layers.



Basic CNN structure.

1D Convolution

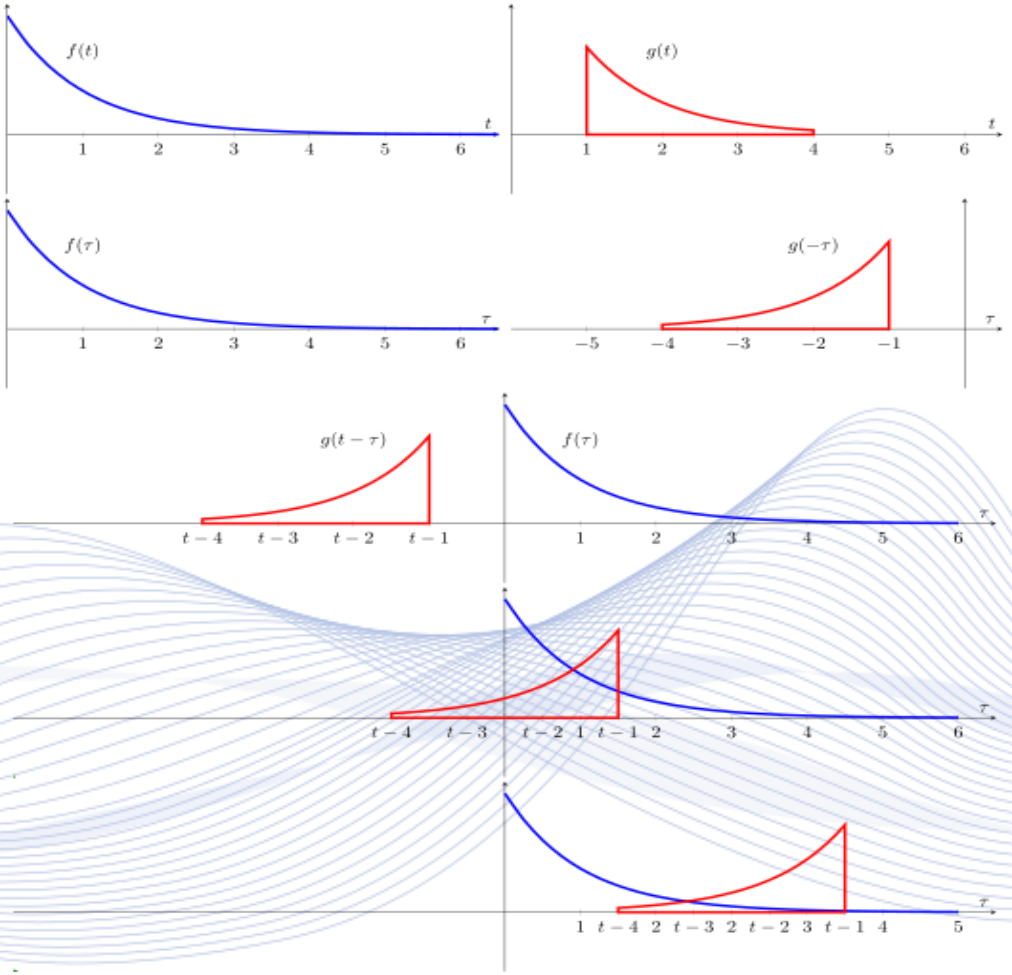
1D (linear) convolution of an M length ***convolution kernel*** (or filter) w with a signal x of size N, is defined by:

$$y(n) = w(n) * x(n) = \sum_{k=0}^{M-1} w(k)x(n-k).$$

- If filter window has odd size $(\mathrm{M} = 2\nu + 1)$ and is centered around 0, 1D convolution takes the form:

$$y(n) = w(n) * x(n) = \sum_{k=-\nu}^{\nu} w(k)x(n-k)\,.$$

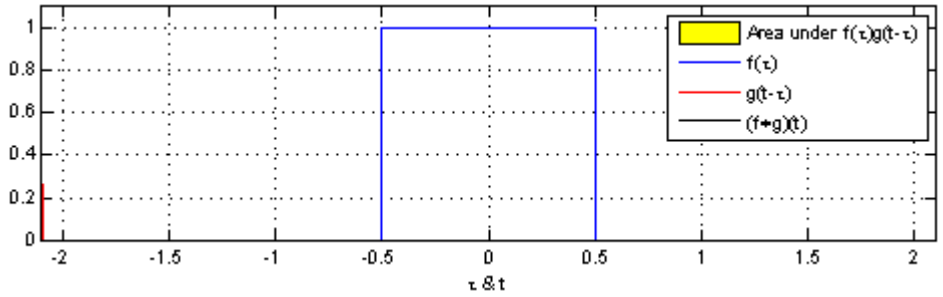
1D Convolution

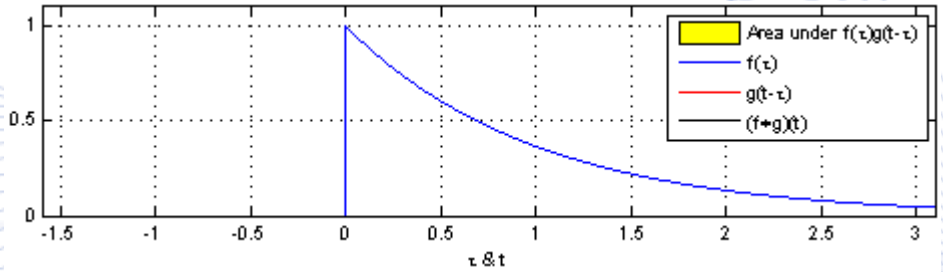


Artificial Intelligence & Information Analysis Lab

Wikipedia contributors, "Convolution," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=Convolution&oldid=1021755990 (accessed May 7, 2021)

1D Convolution



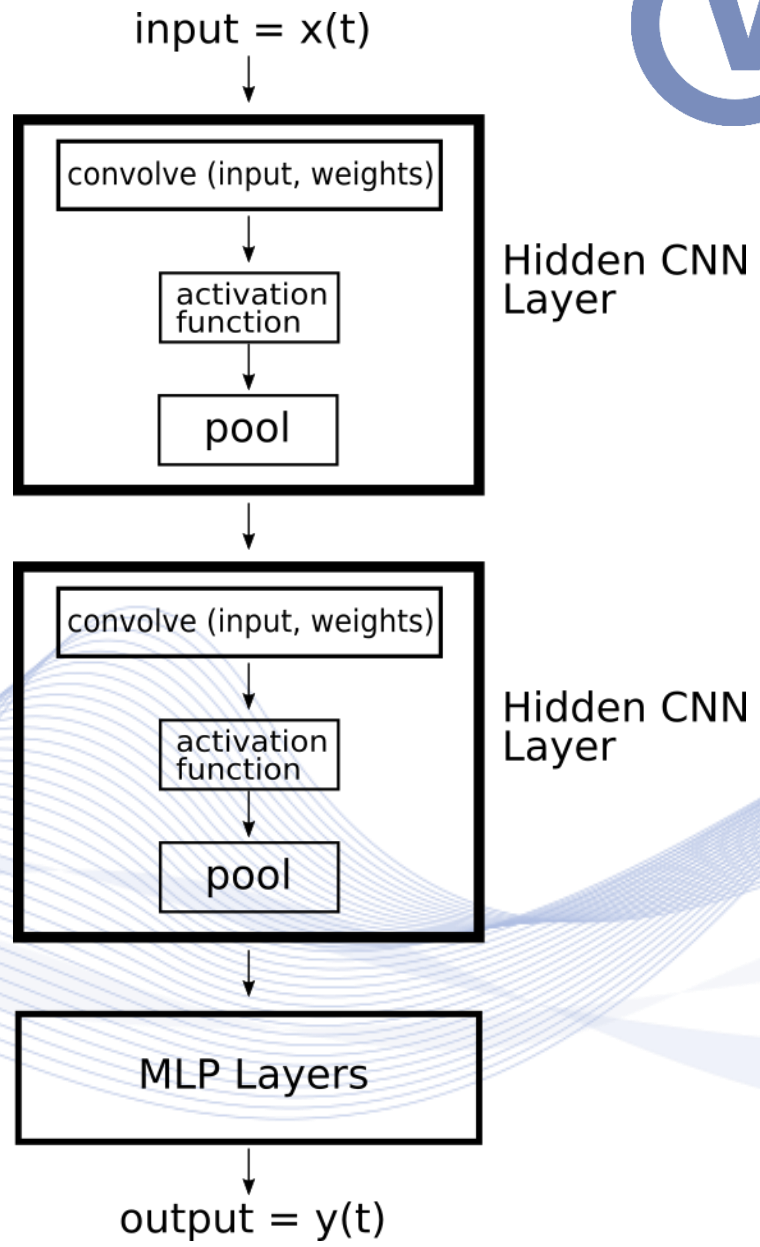


Wikipedia contributors, "Convolution," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=Convolution&oldid=1021755990 (accessed May 7, 2021)

1D CNN Architecture

- Basic skeleton of a 1D CNN.

- Each CNN hidden layer consists of:
  - A convolutional layer with multiple neurons, where each one is a 1D convolutional kernel
  - An activation function
  - A pooling layer for downsampling



Convolutional Layer

Convolutional Layers

- For a convolutional layer l with an activation function $f^{(l)}(\cdot)$, multiple incoming features d_{in} and one single output feature o:

$$a_{io}^{(l)} = f^{(l)}\left(\sum_{r=1}^{d_{in}} \sum_{k=-\nu^{(l)}}^{\nu^{(l)}} w_{kro}^{(l)} a_{(i-k)r}^{(l-1)} + b_o^{(l)}\right).$$

- The input to the first convolutional layer is a multichannel signal x_{jr}:

$$a_{io}^{(0)} = x_{ir}$$

- Feature map $a_{ir}^{(l)}$ downsampling is typically performed, e.g., using ***strided convolutions*** or through ***max/average pooling layers***.

Fully Connected Layer

- The mathematical description of the l^{th} fully connected layer is given by:

$$z_j^{(l)} = \mathbf{w}_j^{(l)T}\mathbf{a}^{(l-1)} + b_j^{(l)},$$

$$a_j^{(l)} = f^{(l)}(\mathbf{w}_j^{(l)T}\mathbf{a}^{(l-1)} + b_j^{(l)}).$$

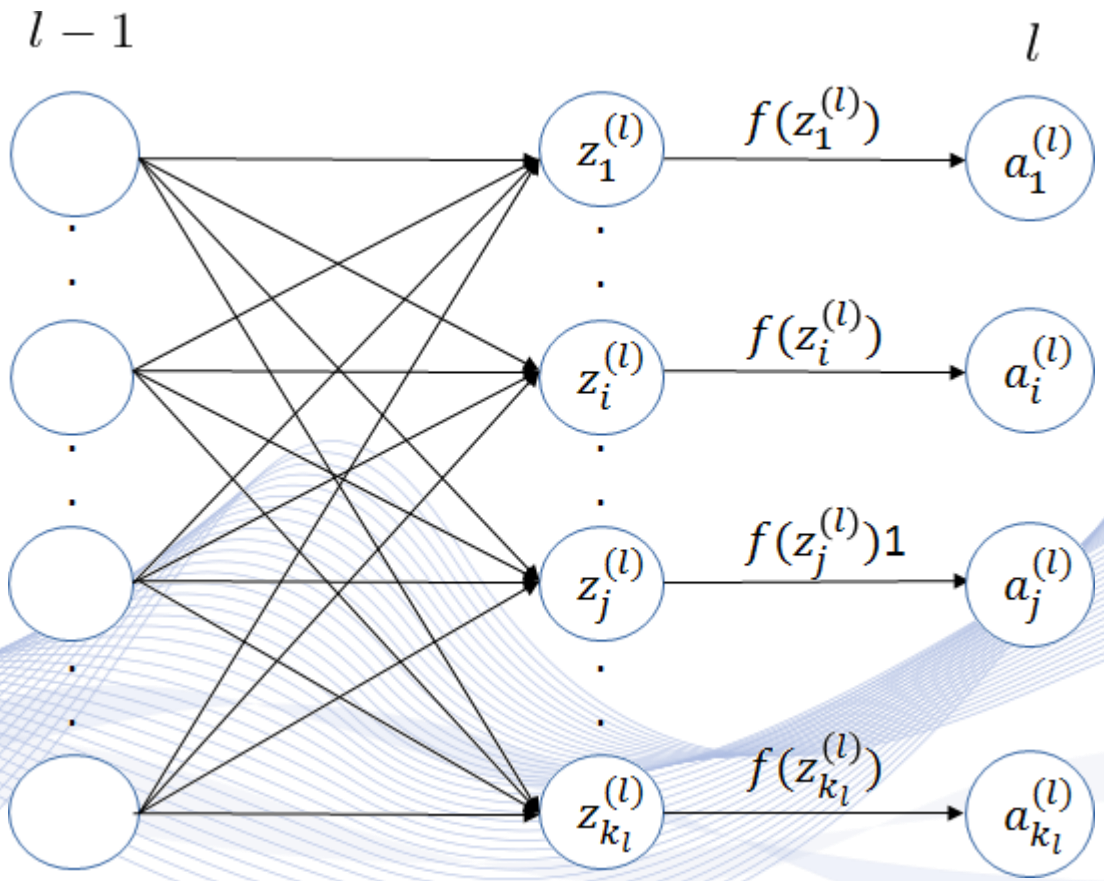
- In compact form:

$$\mathbf{W}^{(l)} = \left[\mathbf{w}_j^{(l)}\right]_{j=1}^{k_l},$$

$$\mathbf{b}^{(l)} = \left[\mathrm{b}_j^{(l)}\right]_{j=1}^{k_l},$$

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)T}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)},$$

$$\boldsymbol{a}^{(l)} = \boldsymbol{f}^{(l)}\left(\mathbf{z}^{(l)}\right).$$



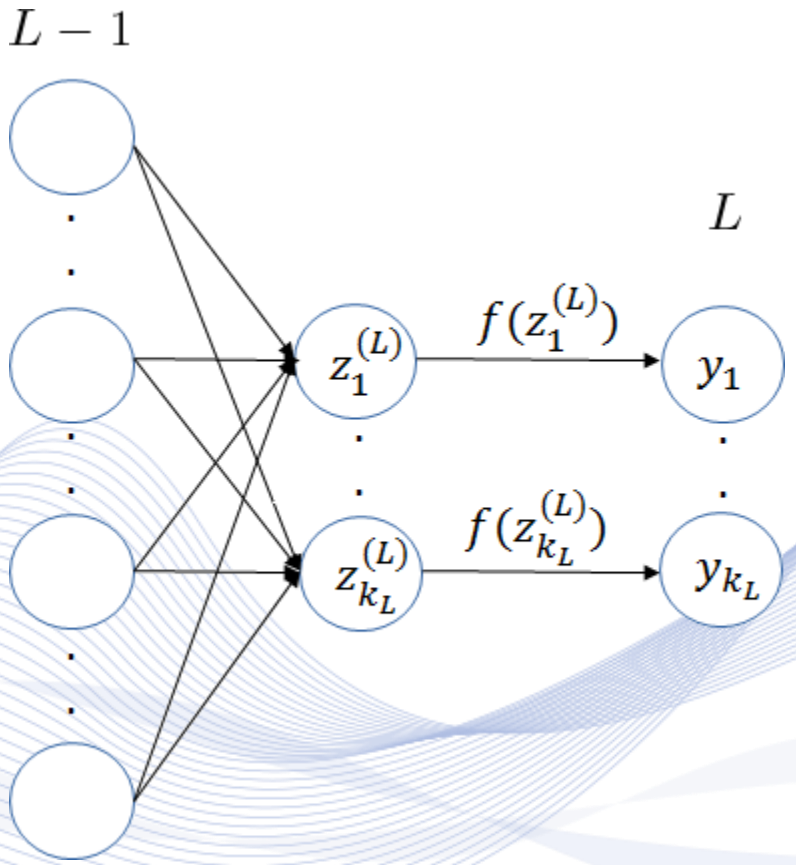
Fully Connected Layer

- Last (output) layer $l = L$, with $k_l = 1$ (regression, two class classification) or m (multi-class classification):

$$z_j^{(L)} = \mathbf{w}_j^{(L)T}\mathbf{a}^{(L-1)} + b_j^{(L)},$$

$$\hat{y}_j = f^{(L)}\left(\mathbf{w}_j^{(L)T}\mathbf{a}^{(L-1)} + b_j^{(L)}\right).$$

- Overall, CNN computes the function $\hat{\mathbf{y}} = \boldsymbol{f}(\mathbf{x}; \boldsymbol{\theta})$, where $\boldsymbol{\theta} \in \mathbb{R}^d$ groups all trainable parameters (convolution kernels, fully connected layer weights and biases).



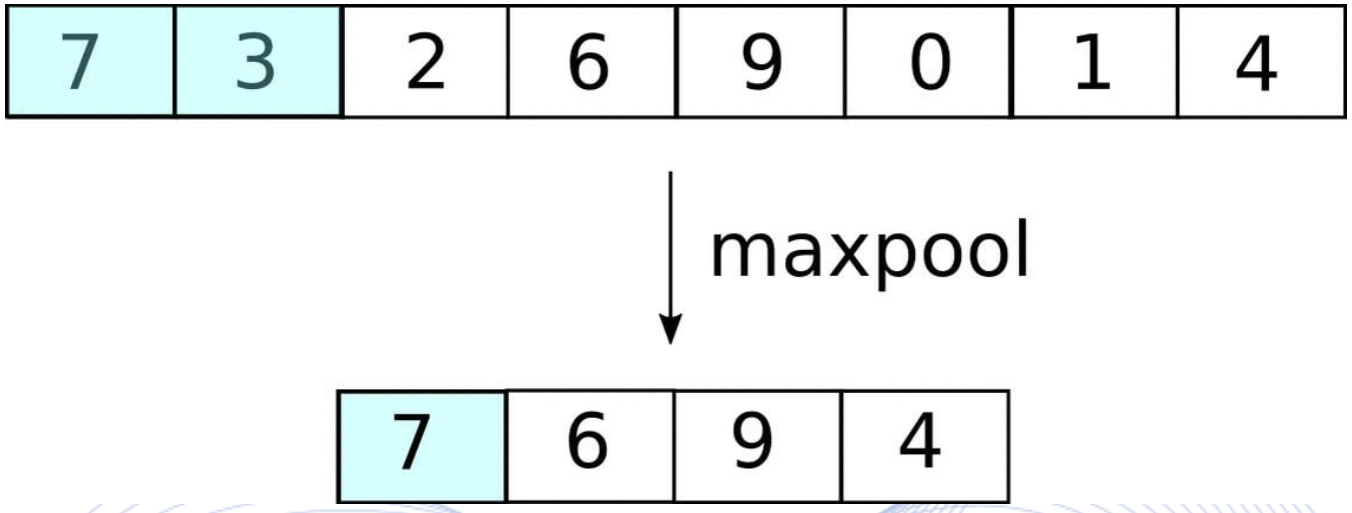
Pooling Layers

Pooling layers are added inside a CNN architecture primarily for downsampling, aiming to reduce the computational cost. Secondarily helps on translation invariance.

- The pooling window is moved over an activation map $\mathbf{A}_{io}^{(l)}$ with stride s.
- Typical pool window sizes $2,\ 3$.
- Downsampling usually with $s = 2$.
- Pools could overlap, e.g., max-pooling window with length 3, stride 2.

- Ad-hoc decision to use pooling or not.
- No formal justification for the effect of overlapping on pooling regions.

Pooling Layers



Activation Functions

- ***Sigmoid*** and ***hyperbolic tangent*** function are not proper for CNNs, because they lead to the *vanishing gradients* problem.

- ***Rectifiers*** are more suitable for activation functions.
  - **ReLU** - Rectified Linear Unit

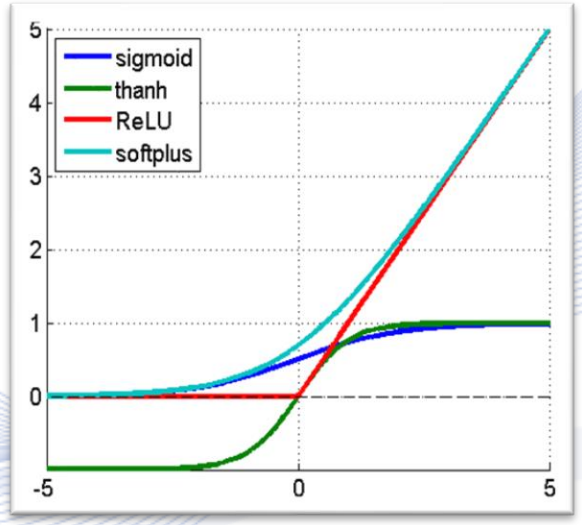
$$\mathrm{a} = ReLU(z) = \max\{\mathrm{z}, 0\} \quad : \mathbb{R} \rightarrow [0, +\infty)$$

  - **ReLU6** - Rectified Linear Unit Bounded by 6

$$\mathrm{a} = min\ \{ReLU(z), 6\} = \min\{\max\{\mathrm{z}, 0\}, 6\} \quad : \mathbb{R} \rightarrow [0,6]$$

  - **Softplus**

$$\mathrm{a} = softplus(z) = log(1 + e^{z}) \quad : \mathbb{R} \rightarrow [0, +\infty)$$

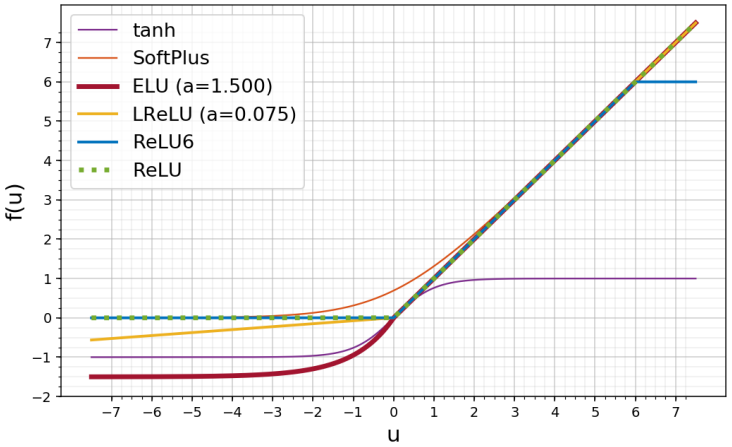


Activation Functions

- Activation functions with positive range of values have mean activation larger than zero, leading to the ***bias shift*** problem.
  - Activations of neurons should be capable of cancelling each other at the next layer.
- Advanced activation functions try to mitigate the negative effects of ReLU.
- ***Leaky Rectified Linear Unit*** (***LReLU***):

$$a = LReLU(z) = \begin{cases} ReLU(z), & z \geq 0 \\ -c \cdot ReLU(-z), & z < 0. \end{cases}$$

- ***Parametric Rectified Linear Unit*** (***PReLU***).
- ***Randomized Leaky Rectified Linear Unit*** (**RReLU**).

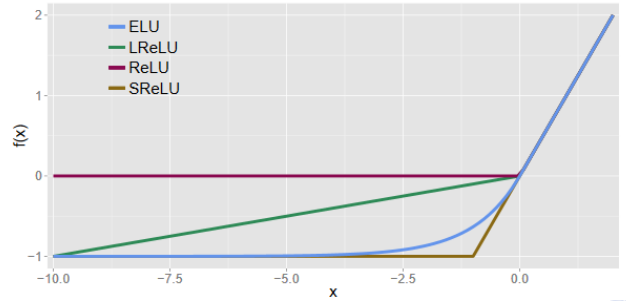


Activation Functions

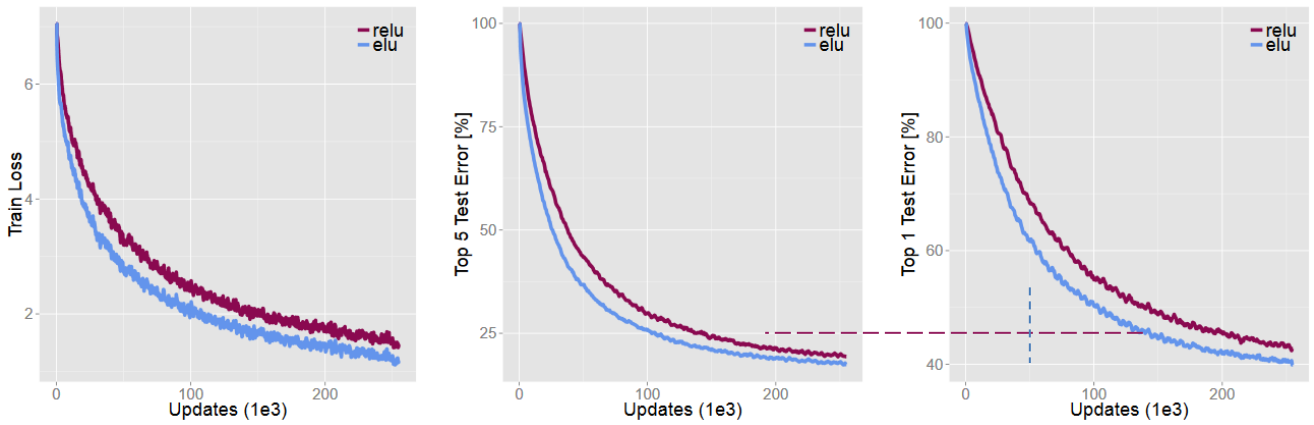
- ***ELU*** activation function aims to mitigate the bias shift problem:

$$a = ELU(z) = \begin{cases} ReLU(z) = \max\{z, 0\}, & z \geq 0 \\ c \cdot (e^z - 1), & z < 0 \end{cases},$$

$$0 < c \leq 1$$



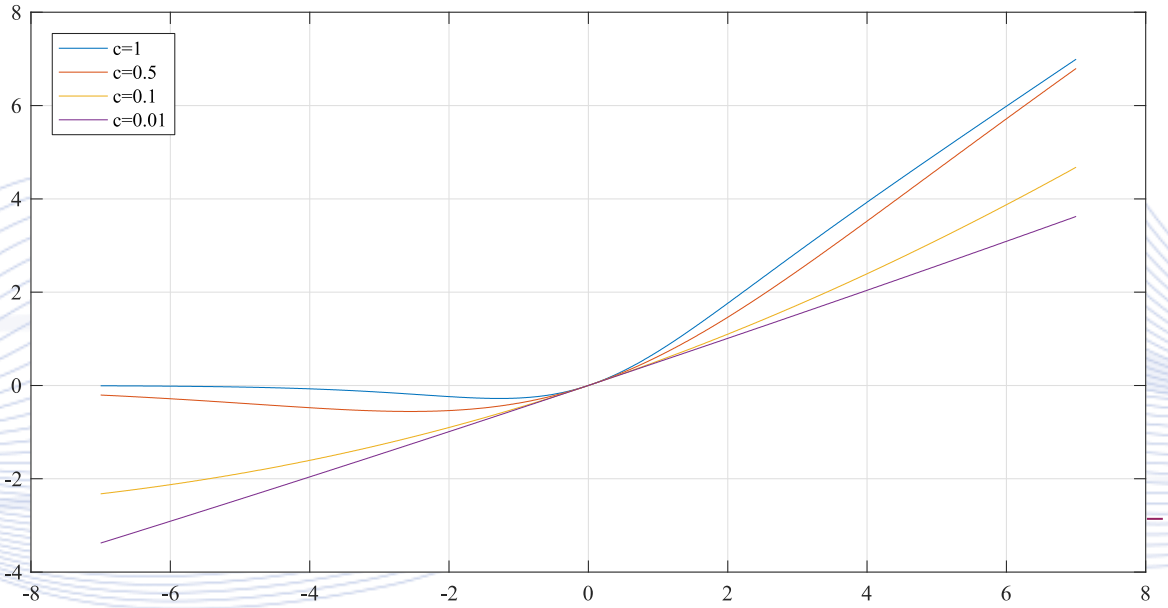
- It achieves faster training convergence.



Activation Functions

- ***Swish*** activation function has been found to provide improved performance for large datasets:

$$a = z\sigma(z) = \frac{z}{1 + e^{-cz}}.$$



Supervised Learning

- A sufficient large training sample set $\mathcal{D}$ is required for Supervised Learning (regression, classification):

$$\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \ldots, N\}.$$

- $\mathbf{x}_i \in \mathbb{R}^n$: $n-$ dimensional input (feature) vector of the i-th training sample.
- $\mathbf{y}_i$: its target label (output).
- Target form $\mathbf{y}$ can vary:
  - it can be categorical, a real number or a combination of both.

Supervised Learning

- ***Training***: Given N pairs of training samples $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \dots N\}$, where $\mathbf{x}_i \in \mathbb{R}^n$ and $\mathbf{y}_i \in [0,1]^m$, estimate $\boldsymbol{\theta}$ by minimizing a loss function: $\min_{\boldsymbol{\theta}} J(\mathbf{y}, \hat{\mathbf{y}})$.

- ***Inference/testing***: Given N_t pairs of testing examples $\mathcal{D}_t = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \dots, N_t\}$, where $\mathbf{x}_i \in \mathbb{R}^n$ and $\mathbf{y}_i \in [0,1]^m$, compute (***predict***) $\hat{\mathbf{y}}_i$ and calculate a performance metric, e.g., classification accuracy.

Classification/Recognition/ Identification

- Given a set of classes $\mathcal{C} = \{\mathcal{C}_i,\ i = 1, \dots, m\}$ and a sample $\mathbf{x} \in \mathbb{R}^n$, the ML model $\hat{\mathbf{y}} = \mathbf{f}(\mathbf{x}; \boldsymbol{\theta})$ predicts a class label vector $\hat{\mathbf{y}} \in [0, 1]^m$ for input sample $\mathbf{x}$, where $\boldsymbol{\theta}$ are the learnable model parameters.

- Essentially, a probabilistic distribution $P(\hat{\mathbf{y}}; \mathbf{x})$ is computed.
- Interpretation: likelihood of the given sample $\mathbf{x}$ belonging to each class $\mathcal{C}_i$.

- Single-target classification:
  - Classes $\mathcal{C}_i, i = 1, \dots, m$ are mutually exclusive: $\|\hat{\mathbf{y}}\|_1 = 1$.
- Multi-target classification:
  - Classes $\mathcal{C}_i, i = 1, \dots, m$ are not mutually exclusive : $\|\hat{\mathbf{y}}\|_1 \geq 1$.

Classification

- **Classification:**
  - Two class ($m = 2$) and multiple class ($m > 2$) classification.
  - **Example:** *Emotion detection from Speech* (two classes), *emotion recognition from Speech* (many classes).

Classification

Multiclass Classification $(m > 2)$**:**

- Multiple $(m > 2)$ hypothesis testing: choose a winner class out of m classes.
- Binary hypothesis testing:
  - One class against all: m binary hypothesis.
    - one must be proven true.
  - Pair-wise class comparisons: $m(m-1)/2$ binary hypothesis.

Regression

Given a sample $\mathbf{x} \in \mathbb{R}^n$ and a function $\mathbf{y} = \boldsymbol{f}(\mathbf{x})$, the model predicts ***real-valued quantities*** for that sample: $\hat{\mathbf{y}} = \boldsymbol{f}(\mathbf{x}; \boldsymbol{\theta})$, where $\hat{\mathbf{y}} \in \mathbb{R}^m$ and $\boldsymbol{\theta}$ are the learnable parameters of the model.

- **Training**: Given N pairs of training examples $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \dots N\}$, where $\mathbf{x}_i \in \mathbb{R}^n$ and $\mathbf{y}_i \in \mathbb{R}^m$, estimate $\boldsymbol{\theta}$ by minimizing a loss function: $\min_{\boldsymbol{\theta}} J(\mathbf{y}, \hat{\mathbf{y}})$.

- **Testing**: Given N_t pairs of testing examples $\mathcal{D}_t = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \dots, N_t\}$, where $\mathbf{x}_i \in \mathbb{R}^n$ and $\mathbf{y}_i \in \mathbb{R}^m$, compute (predict) $\hat{\mathbf{y}}_i$ and calculate a performance metric, e.g., MSE.

CNN Training

- ***Mean Square Error (MSE)***:

$$J(\boldsymbol{\theta}) = J(\mathbf{W}, \mathbf{b}) = \frac{1}{N}\sum_{i=1}^{N} \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|^2.$$

  - It is suitable for regression and classification.

- ***Categorical Cross Entropy Error*:**

$$J_{CCE} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i).$$

  - It is suitable for classifiers that use softmax output layers.

CNN Training

- ***Exponential Loss***:

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{N} e^{-\beta \mathbf{y}_i^T \hat{\mathbf{y}}_i}.$$

- ***Hinge Loss:***

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{N} \max(0, 1 - y_i \hat{y}_i)\,.$$

- It can be used with binary classification where the target values are in the set $\{-1, 1\}$.

Backpropagation

- The most widespread algorithm for supervised training of CNNs is the Backpropagation algorithm.
- Unlike typical MLPs, the summation that occurs inside each convolutional neuron during forward pass uses convolution instead of normal multiplication.
- Training is performed as a problem of minimization of a cost function.
- MSE is the most common cost function.

CNN Training

- CNNs are trained with the same gradient descent methods as multilayer perceptron.

- Optimization methods:
  - ***Learning rate decay*** allows scheduled changes to the learning rate at the various training epochs.

  - ***ADAM*** is an optimization method with an adaptive learning rate.

- Large scale datasets are needed to adequately train a CNN.

CNN Training

Data Augmentation:

- It is used to avoid overfitting.
- The training set is augmented during training with label-preserving transformation of the samples.

CNN Training

Softmax Layer:

- It is the last layer in several neural network classifiers.
- The response of neuron i in the softmax layer L is calculated with regard to the value of its activation function $a_i = f(z_i)$:

$$\hat{y}_i = g(a_i) = \frac{e^{a_i}}{\sum_{k=1}^{k_L} e^{a_k}} : \mathbb{R} \to [0,1], \qquad i = 1,..,k_L,$$

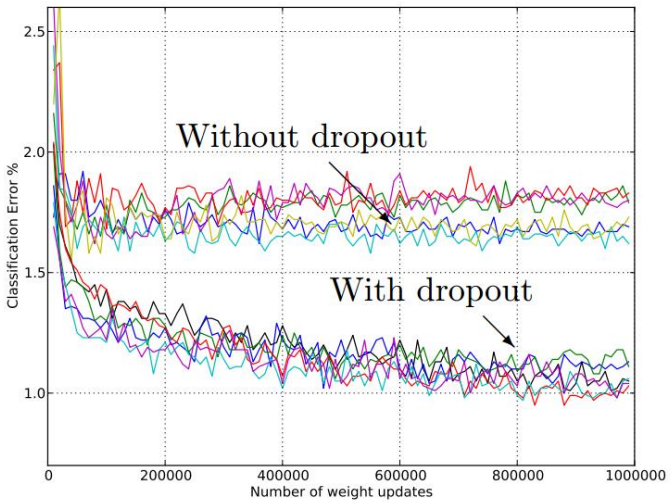
- The responses of softmax neurons sum up to one: $\sum_{i=1}^{k_L} \hat{y}_i = 1$.

- Better representation for mutually exclusive class labels.

CNN Training

Dropout randomly excludes a set of neurons from a certain training epoch with a constant keep out probability p_{keep}.

- Activations of dropped out neurons are set to zero.
  - They do not participate in the loss, thus excluded from back-propagation.
  - Dropout was initially used in AlexNet after each fully connected layer.
  - During testing a trained model, all neurons are used with their already learned weights.
- Induces dynamic sparsity during training.
- Prevents complex co-adaptations of the synaptic weights, that may lead to correlated activations of neurons.



1D CNN Use Cases Examples

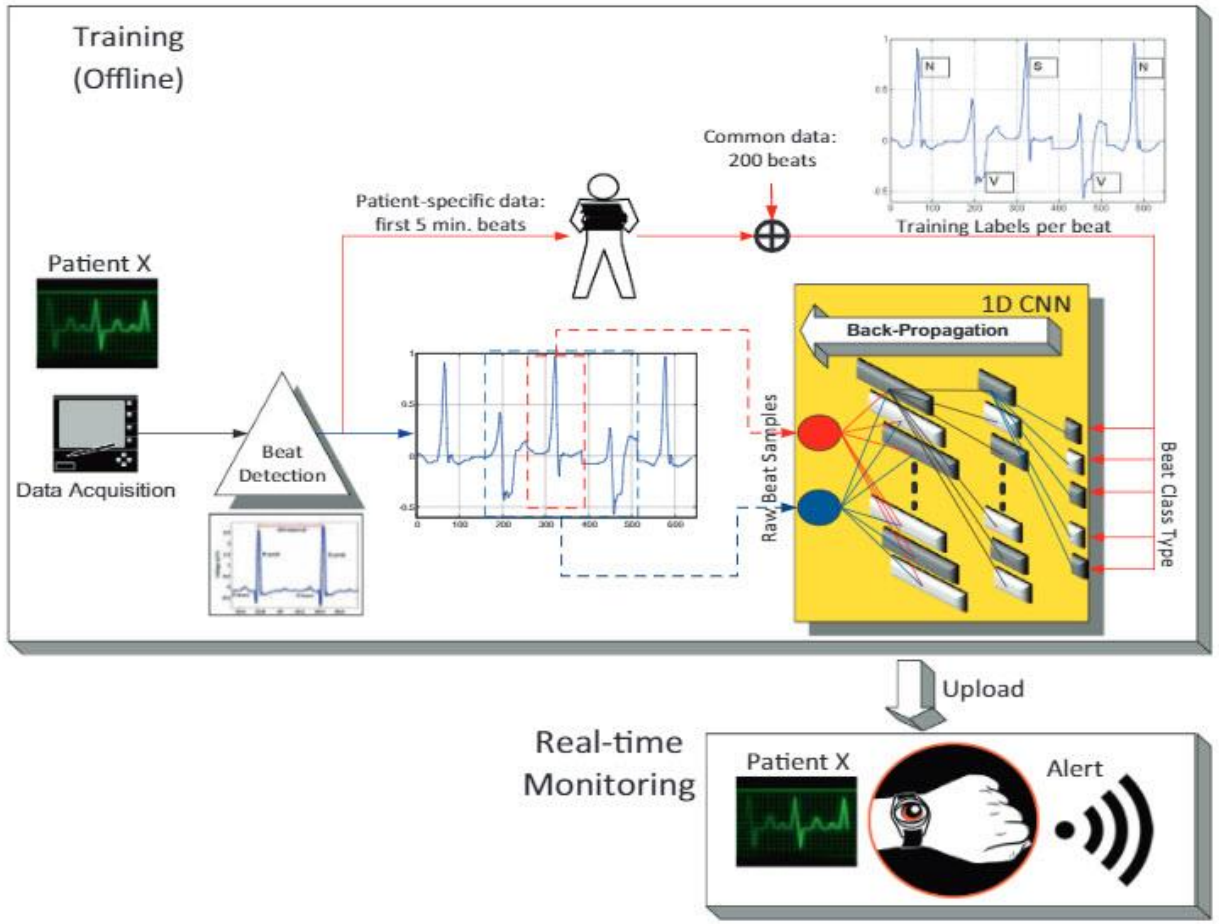
- ECG Monitoring
- Music Auto - Tagging

ECG Monitoring

- Because of high inter-patient variations of ECG signals, the multiple methods that proposed a single classifier algorithm, didn’t have much success.

- For example, patients that may have arrhythmia have different normal heart beats, that may however, be extremely similar to another patient’s abnormal beats.

ECG Monitoring



Overview of the arrythmia detection and identification system proposed on [real-time patient specific ECG classification]. Image from [1]

ECG Monitoring

- Modelling common causes of cardiac arrhythmia in the signal domain by a degrading system.

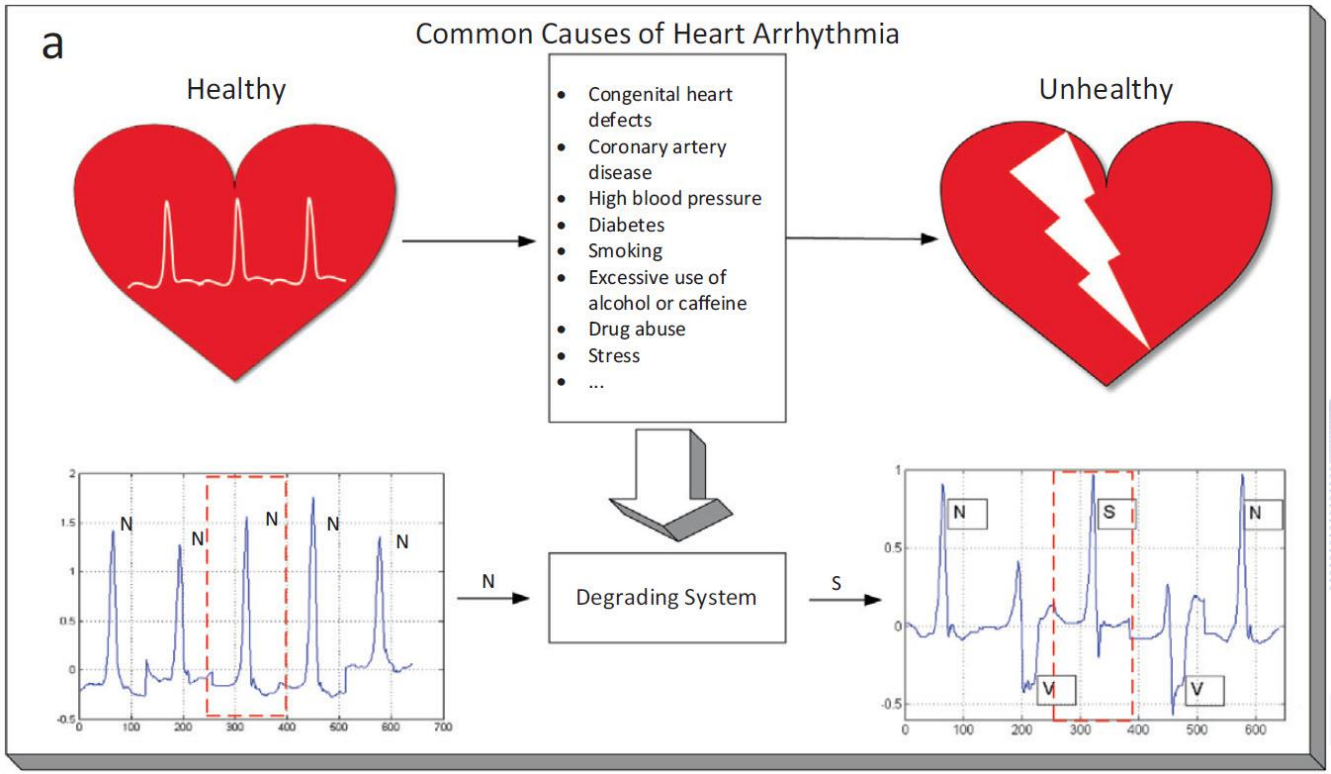


Image from [Personalized monitoring and advance warning system for cardiac arrhythmias]

ECG Monitoring

- An illustration of an abnormal S beat synthesis.

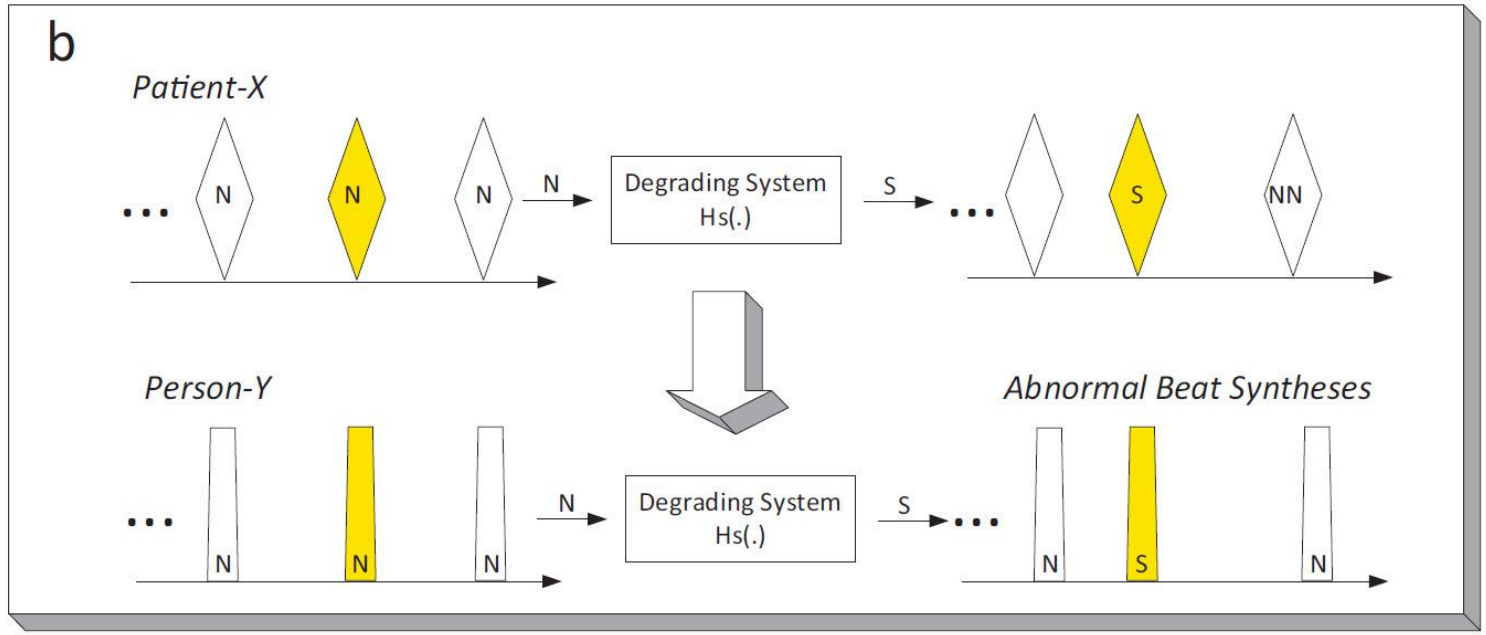


Image from [Personalized monitoring and advance warning system for cardiac arrhythmias]

ECG Monitoring

- Illustration of the overall system. Once the 1D CNN is trained, it can be used as a continuous cardiac monitoring and warning system.

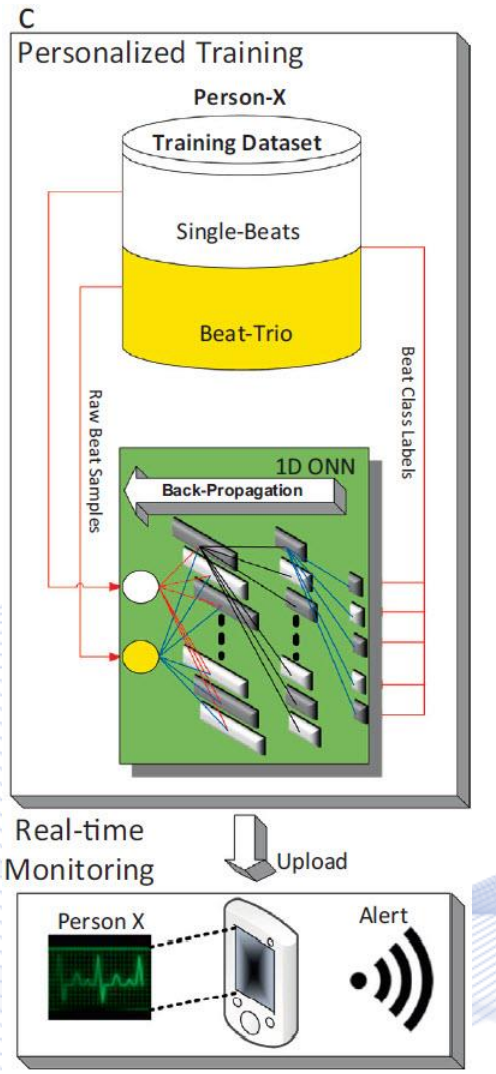
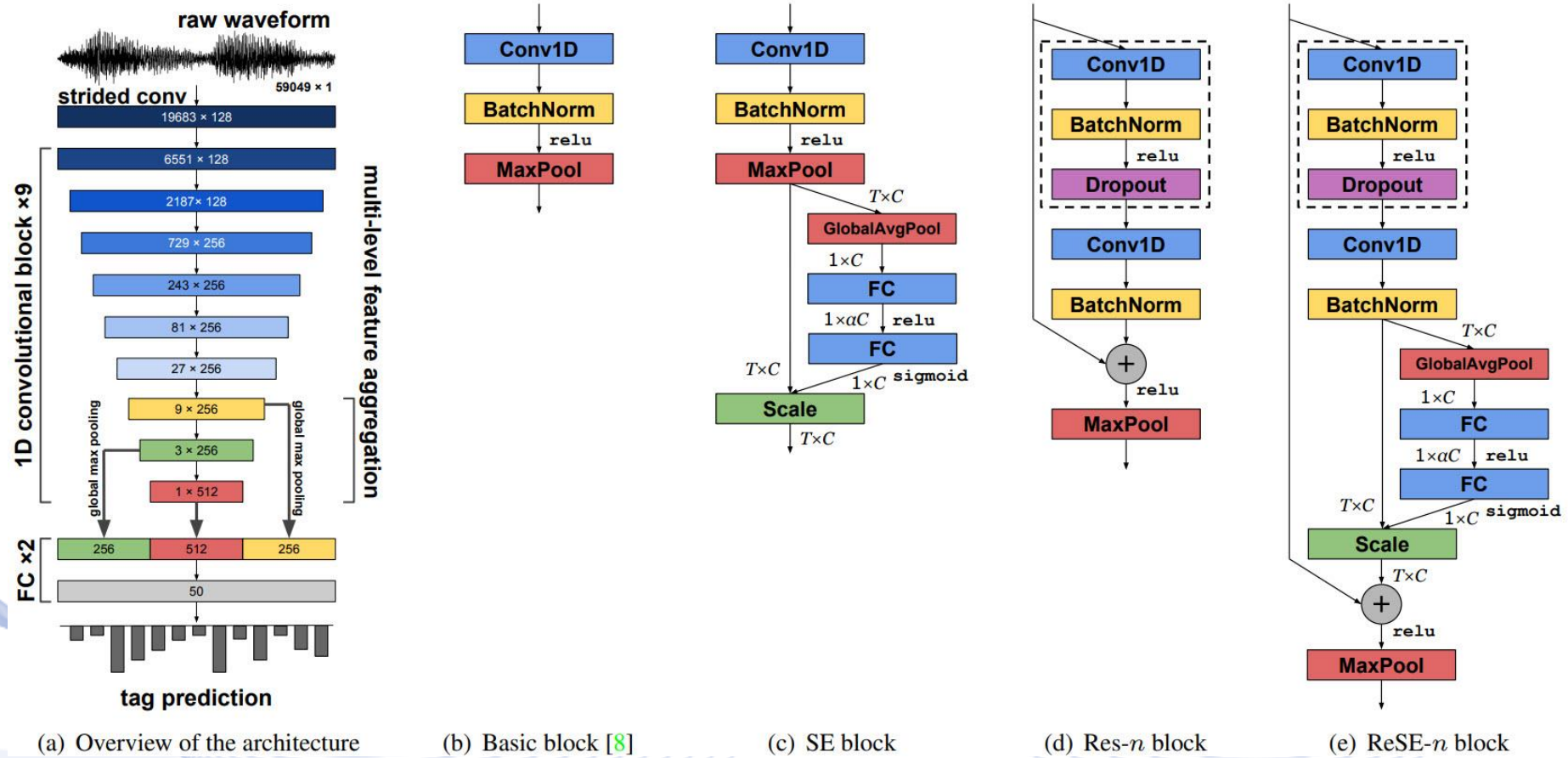


Image from [Personalized monitoring and advance warning system for cardiac arrhythmias]

Music Auto - Tagging

- In [sample-level architectures for music auto-tagging using raw waveforms] a 1-D CNN architecture was proposed for music auto-tagging.
- This network adopted building blocks from image classification models (ResNets and SENets).
- The results show significant improvements on accuracy in MagnaTagATune dataset, and comparable results on Million Song Dataset.

Music Auto - Tagging



(a) Overview of the architecture (b) Basic block [8] (c) SE block (d) Res-n block (e) ReSE-n block

Image from [sample-level architectures for music auto-tagging using raw waveforms]

Bibliography

[1] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, Daniel J. Inman, “1D convolutional neural networks and applications: A survey”,Mechanical Systems and Signal Processing,Volume 151,2021,107398,ISSN 0888-3270

[2] Wikipedia contributors, "Convolution," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=Convolution&oldid=1022204425 (accessed May 9, 2021).

[3] LeCun Y, Bengio Y, Hinton G. ‘Deep learning’ Nature, 2015 May;521(7553):436-44.

[4] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016

[5] Schmidhuber J. Deep learning in neural networks: An overview. Neural networks. 2015 Jan 1;61:85-117.

Bibliography

[6] R. Miikkulainen et al. "Evolving deep neural networks." Artificial Intelligence in the Age of Neural Networks and Brain Computing. Academic Press, 2019. 293-312.

[7] A. Canziani, A. Paszke, and E. Culurciello, “An Analysis of Deep Neural Network Models for Practical Applications,” arXiv:1605.07678 [cs], May 2016.

[8] Serkan Kiranyaz, Turker Ince, and Moncef Gabbouj, “Real-time patient-specific ecg classification by 1-d convolutional neural networks,”IEEETransactions on Biomedical Engineering, vol. 63, no. 3, pp. 664–675,2016

Bibliography

[9] Ince T. Kiranyaz S. and M. Gabbouj, “Personalized monitoring andadvance warning system for cardiac arrhythmias,”Sci Rep 7, 2017.

[10] Taejun Kim, Jongpil Lee, and Juhan Nam, “Sample-level cnn architectures for music auto-tagging using raw waveforms,” in2018 IEEEInternational Conference on Acoustics, Speech and Signal Processing(ICASSP), 2018, pp. 366–370.

[11] Wikipedia contributors, “Convolutional neural network — Wikipedia,the free encyclopedia,” 2021, [Online; accessed 4-June-2021].

[12] Wikipedia contributors, “Ape tag — Wikipedia, the free encyclopedia,”2021, [Online; accessed 4-June-2021]

Bibliography

[PIT2021] I. Pitas, “Computer vision”, Createspace/Amazon, in press.
[PIT2017] I. Pitas, “Digital video processing and analysis” , China Machine Press, 2017 (in Chinese).
[PIT2013] I. Pitas, “Digital Video and Television” , Createspace/Amazon, 2013.
[NIK2000] N. Nikolaidis and I. Pitas, “3D Image Processing Algorithms”, J. Wiley, 2000.
[PIT2000] I. Pitas, “Digital Image Processing Algorithms and Applications”, J. Wiley, 2000.

Q & A

Thank you very much for your attention!

More material in
http://icarus.csd.auth.gr/cvml-web-lecture-series/

Contact: Prof. I. Pitas
pitas@csd.auth.gr