FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

AN EMPITICAL EVALUATION OF GENERIC CONVOLUTIONAL AND RECURRENT NETWORKS FOR SEQUENCE MODELING

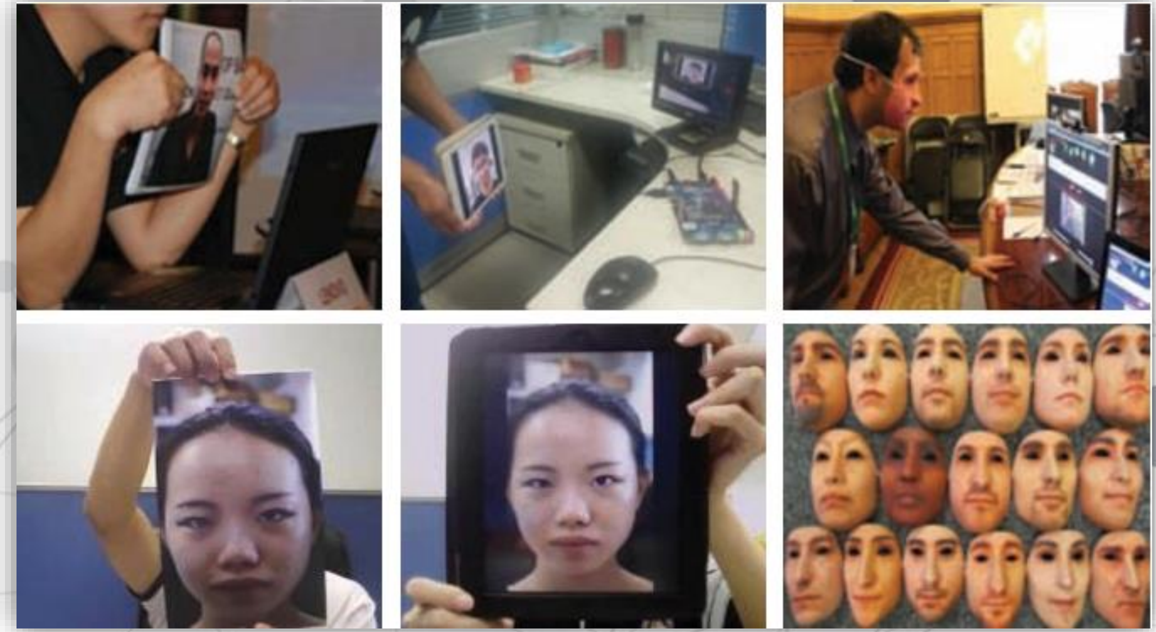
Nova Search Reading Group – June 4th 2020

Ruslan Padnevych

Shaojie Bai, J. Zico Kolter and Vladlen Koltun. April 2018

https://arxiv.org/pdf/1803.01271.pdf

IDENTITY THEFT



Different identity theft techniques

IDENTITY THEFT (CONT.)

In 2011, a passenger boarded the plane in Hong Kong wearing an elderly man's mask and successfully landed in Canada.

Photos taken from Google

CONTEXT: BORDERS WITHOUT BARRIERS

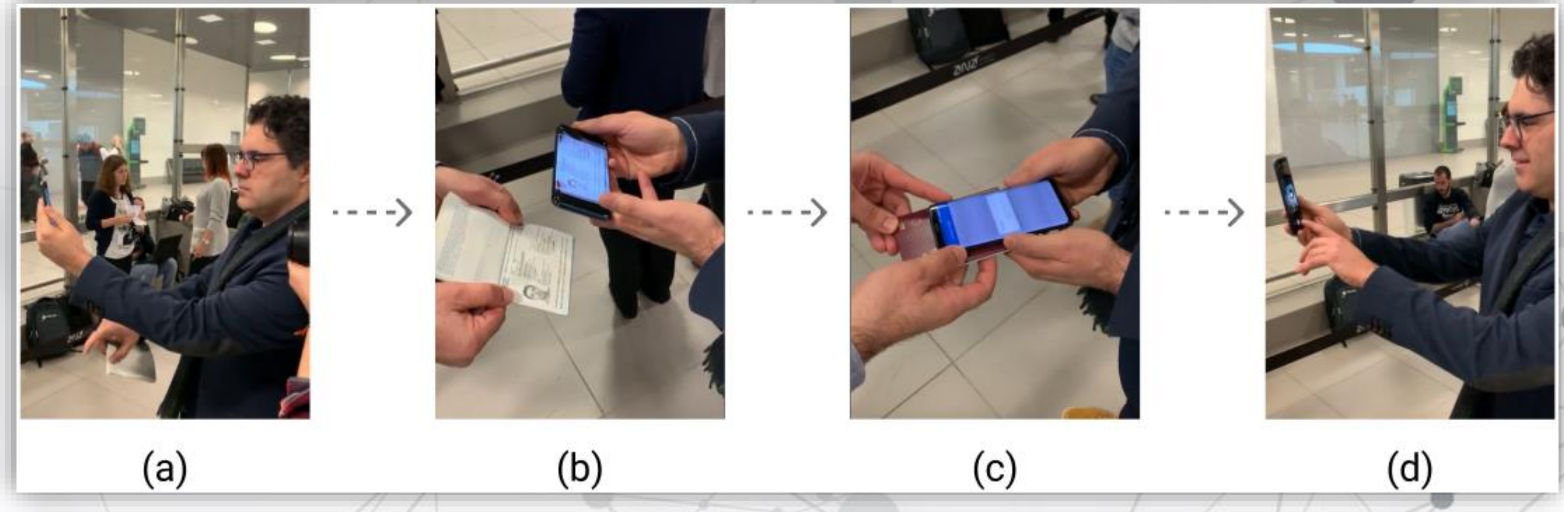
Use case

Biometrics on the Move (SmartyFlow project)

Queues to show documents to border guards

CONTEXT: BORDERS WITHOUT BARRIERS

Registration Phase

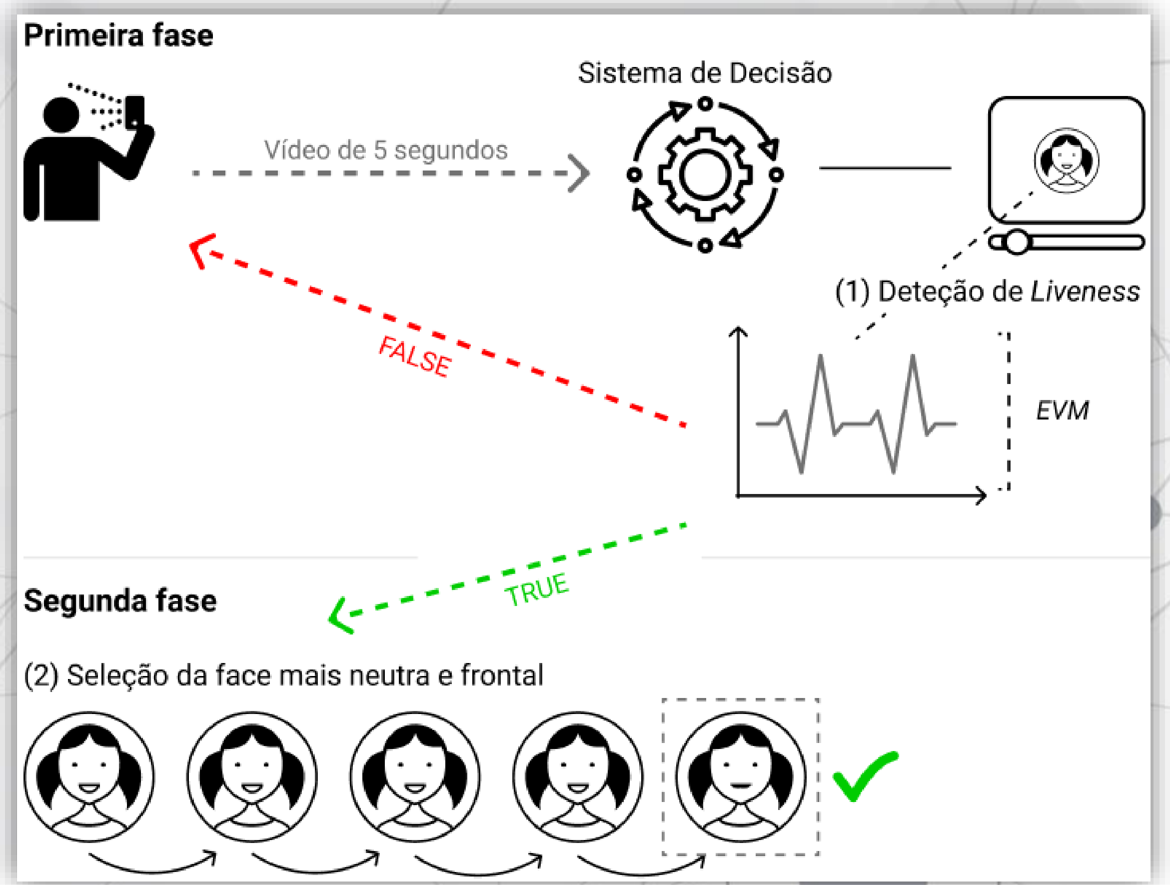


(a) → liveliness detection (5s video)
(b) - passport registration
(c) - reading the passport chip
(d) - verification of correspondence

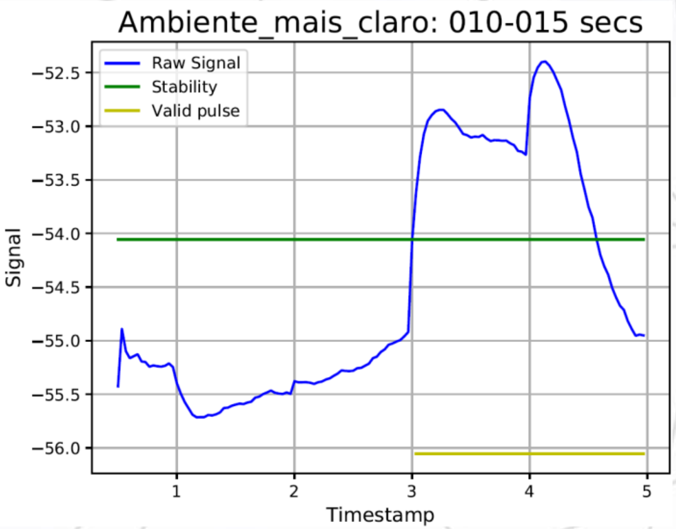
GOAL

Liveliness detection

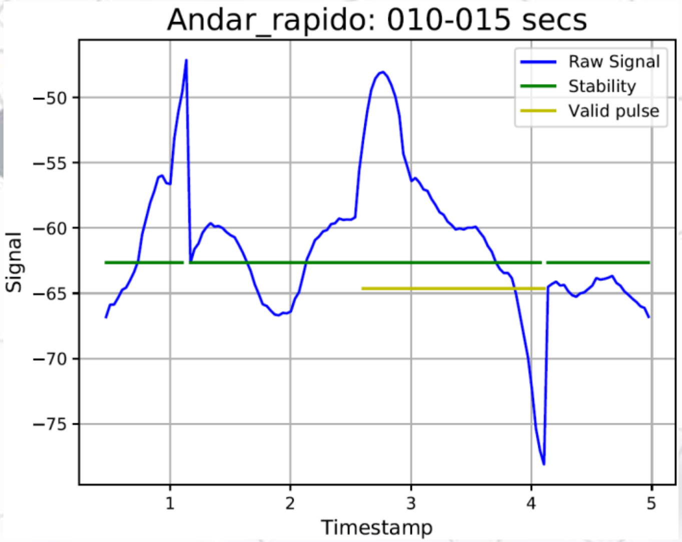
Neutral face selection



Fake



Real



SEQUENCE MODELING

- Suppose that we are given an input sequence x0, ….., xT , and wish to predict some corresponding outputs y0, ….., yT at each time.
- The key constraint is that to predict the output yt for some time t, we are constrained to only use those inputs that have been previously observed: x0, ….., xt.
- Formally, a sequence modeling network is any function:

$$f : \mathcal{X}^{T+1} \to \mathcal{Y}^{T+1}$$

that produces the mapping

$$\hat{y}_0, \ldots, \hat{y}_T = f(x_0, \ldots, x_T)$$

yt depends only on x0, ….., xt and not on any "future" inputs xt+1, ….., xT

SEQUENCE MODELING (CONT.)

The goal of learning in the sequence modeling setting is to find a network f that minimizes some expected loss between the actual outputs and the predictions,

L(y0, ….., yT ; f(x0, ….., xT)),

where the sequences and outputs are drawn according to some distribution.

BACKGROUND

Convolutional Networks

- Used prominently for speech recognition in the 80s and 90s
- Applied to NLP tasks such as part-of-speech tagging and semantic role labelling
- Applied to sentence and document classification
- More recently, applied to **language modeling and machine translation**

Recurrent Networks

- Gained tremendous popularity due to prominent applications to **language modeling and machine translation**

Should Recurrent Networks be regarded as a natural starting point for sequence modeling tasks?

RECURRENT NETWORK (RNN)

- RNN is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence (Rumelhart, 1988).
- Unlike feedforward neural networks, RNNs contain cycles and use an **internal state memory** ***h*** to process sequences of inputs.
- A basic recurrent neural network is described by the propagation equations:

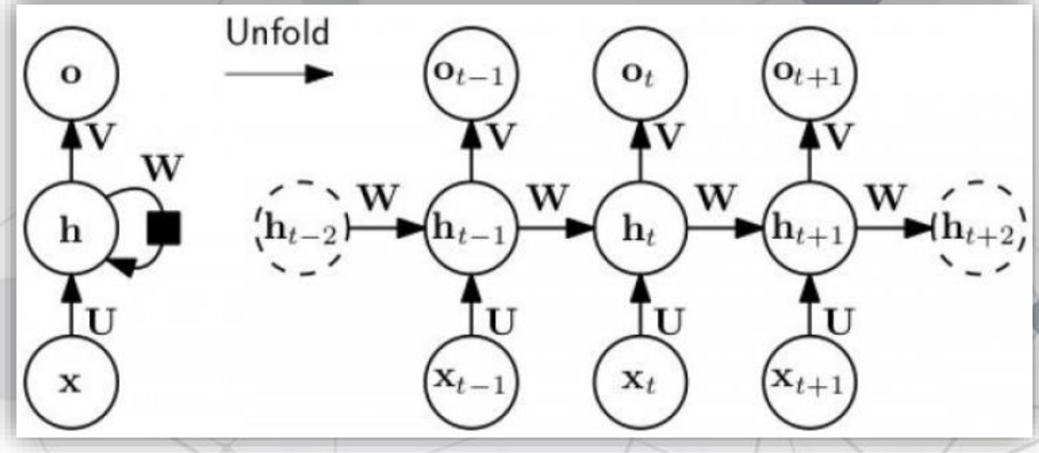
$$\mathbf{h}_t = \sigma(\mathbf{U} \cdot \mathbf{x}_t + \mathbf{W} \cdot \mathbf{h}_{t-1} + \mathbf{b})$$
$$\mathbf{o}_t = \mathbf{V} \cdot \mathbf{h}_t + \mathbf{c}$$

- Where the parameters are the bias vectors **b** and **c** along with the weight matrices:
  - **U** – input-to-hidden
  - **V** – hidden-to-output
  - **W** – hidden-to-hidden

https://dida.do/blog/machine-learning-approaches-for-time-series

RNN (CONT.)

- The computational graph and its unfolded version is shown in the following figure:



- Weight matrices:
  - **U** – input-to-hidden
  - **V** – hidden-to-output
  - **W** – hidden-to-hidden

- Computing the gradients involves performing a forward propagation pass through the unrolled graph followed by a backward propagation pass.

- The runtime is O(T) and cannot be reduced by parallelization because the forward propagation graph is inherently sequential, i.e., each time step may be computed only after the previous one.

https://dida.do/blog/machine-learning-approaches-for-time-series

RNN (CONT.)

- Recurrent models construct very deep computational graphs by repeatedly applying the same operation at each time step of a long temporal sequence. This gives rise to **the vanishing gradient problem** and makes it notoriously difficult to train RNNs.

- To prevent these difficulties more elaborate recurrent architectures were developed, such as the long short-term memory (LSTM) (Hochreiter, 1997) and the gated recurrent unit (GRU) (Cho, 2014).

https://dida.do/blog/machine-learning-approaches-for-time-series

TEMPORAL CONVOLUTIONAL NETWORK (TCN)

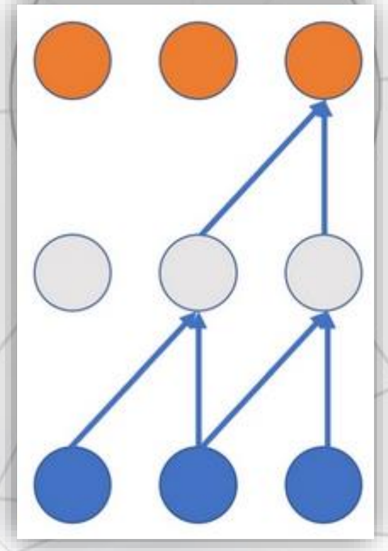
- TCN is inspired by recent convolutional architectures for sequential data and combines simplicity , autoregressive prediction, and very long memory.
- The distinguishing characteristics of TCNs are:

1) The **convolutions in the architecture are causal**, meaning that there is no information "leakage" from future to past.

2) The architecture can take a sequence of any length and map it to an output sequence of the same length, just as with an RNN.

Autoregressive prediction – when we try to predict some signal given its past.

TCN – CAUSAL CONVOLUTIONS



1) TCN uses **causal convolutions,** i.e., convolutions where an output at time t is convolved only with elements from time t and earlier in the previous layer.

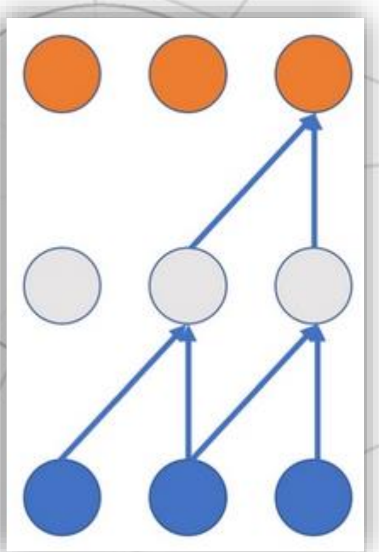
2) TCN uses a **1D fully-convolutional network (FCN)** architecture, where each hidden layer is the same length as the input layer.

TCN = 1D FCN + causal convolutions

TCN (CONT.)

A major disadvantage of this basic design is that in order to achieve a long effective history size, we need an extremely deep network or very large filters.

TCN – DILATED CONVOLUTIONS



- A simple causal convolution is only able to look back at a history with size linear in the depth of the network.

Solution

To employ dilated convolutions that enable an exponentially large receptive field.

TCN – DILATED CONVOLUTIONS

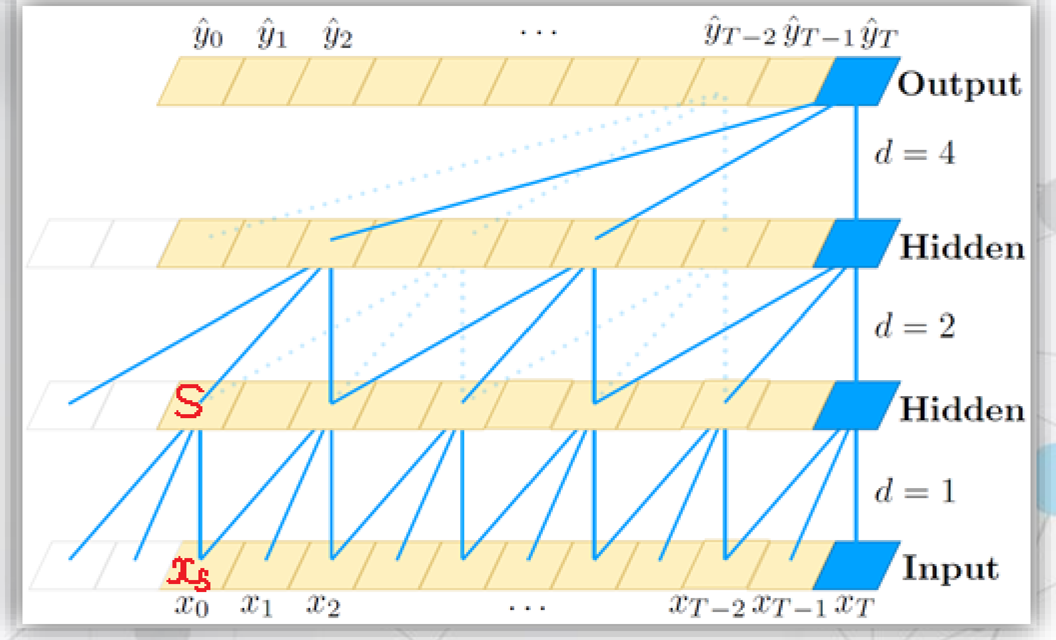
Given a 1-D sequence input $x \in R^n$ and a filter $f : \{f0, \ldots, k-1\} \rightarrow R$, the dilated convolution operation *F* on element *s* of the sequence is defined as:

$$F(s) = (\mathbf{x} *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot \mathbf{x}_{s-d \cdot i}$$

$d = 2^v$ – dilation factor, with *v* the level of the network

k – filter size

(s – d*i) – direction of the past



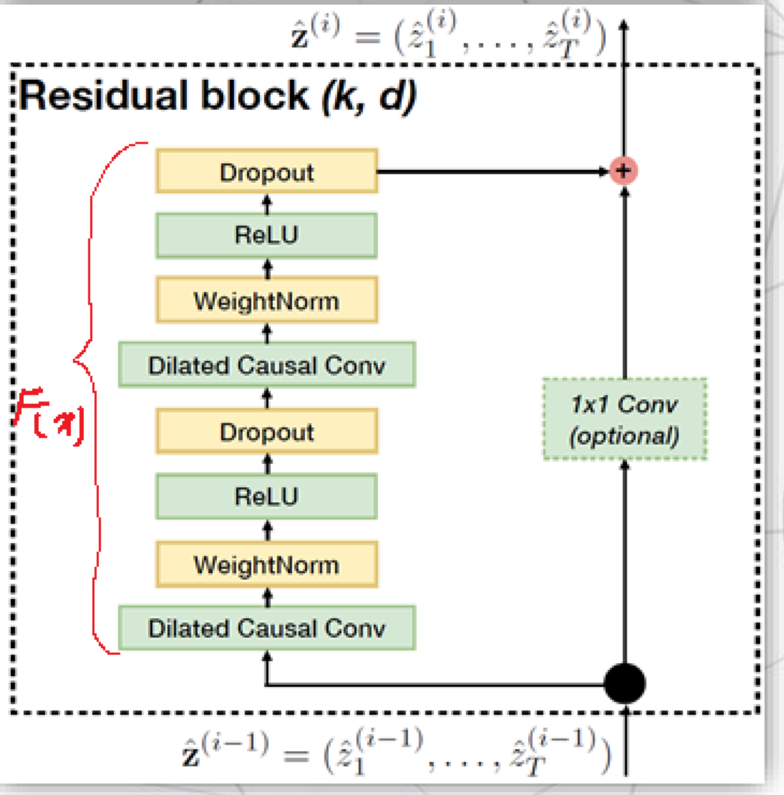
k (filter size) = 3

TCN – RESIDUAL (BLOCKS) CONNECTIONS

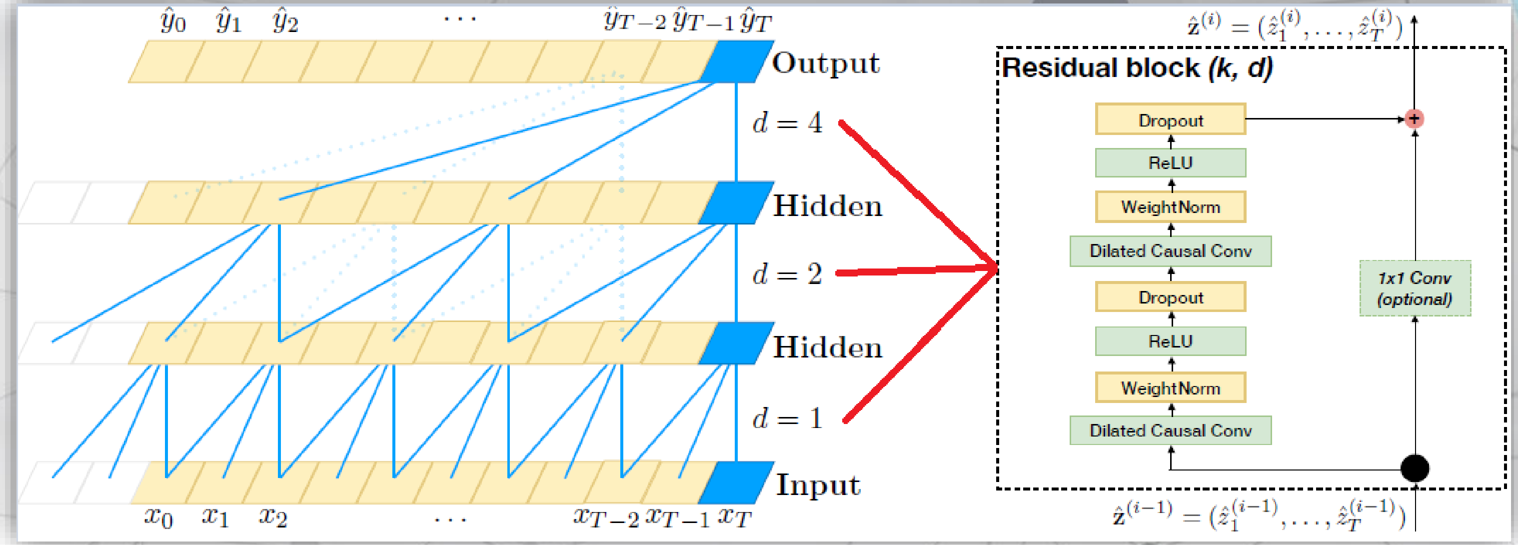
- A residual block contains a branch leading out to a series of transformations *F*, whose outputs are added to the input x of the block:

$$o = \text{Activation}(\mathbf{x} + \mathcal{F}(\mathbf{x}))$$

- This effectively allows layers to learn modifications to the identity mapping *x* rather than the entire transformation *F*(*x*), which has repeatedly been shown to benefit very deep networks.



TCN – RESIDUAL (BLOCKS) CONNECTIONS (CONT.)



- **Normalization:** applied weight normalization to the convolutional filters.
- **Regularization:** a spatial dropout was added after each dilated convolution and at each training step, a whole channel is zeroed out.
- An **1x1 convolution** is added when residual input and output have different dimensions.

TCN – SUMMARY

A simple **temporal convolutional network (TCN)** that combines best practices such as **dilations** and **residual connections** with the **causal convolutions** needed for autoregressive prediction.

TCN HYPERPARAMETERS SETTING

TCN SETTINGS

Dataset/Task	Subtask	k	n	Hidden	Dropout	Grad Clip	Note
The Adding Problem	$T = 200$	6	7	27	0.0	N/A	
	$T = 400$	7	7	27			
	$T = 600$	8	8	24			
Seq. MNIST	-	7	8	25	0.0	N/A	
		6	8	20			
Permuted MNIST	-	7	8	25	0.0	N/A	
		6	8	20			
Copy Memory Task	$T = 500$	6	9	10	0.05	1.0	RMSprop 5e-4
	$T = 1000$	8	8	10			
	$T = 2000$	8	9	10			
Music JSB Chorales	-	3	2	150	0.5	0.4	
Music Nottingham	-	6	4	150	0.2	0.4	
Word-level LM	PTB	3	4	600	0.5	0.4	Embed. size 600
	Wiki-103	3	5	1000	0.4		Embed. size 400
	LAMBADA	4	5	500			Embed. size 500
Char-level LM	PTB	3	3	450	0.1	0.15	Embed. size 100
	text8	2	5	520			

- The most important factor for picking parameters is to make sure that the TCN has a sufficiently large receptive field by choosing *k* and *d* that can cover the amount of context needed for the task.

- There are two ways to increase the receptive field of a TCN: choosing lager filter sizes k and increasing the dilation factor d, since the effective history of one layer is (k-1)*d.

RESULTS

^{h} means that higher is better.

$^\ell$ means that lower is better.

Sequence Modeling Task	Model Size (≈)	Models			
		LSTM	GRU	RNN	**TCN**
Seq. MNIST (accuracy^h)	70K	87.2	96.2	21.5	**99.0**
Permuted MNIST (accuracy)	70K	85.7	87.3	25.3	**97.2**
Adding problem T=600 (loss$^\ell$)	70K	0.164	**5.3e-5**	0.177	**5.8e-5**
Copy memory T=1000 (loss)	16K	0.0204	0.0197	0.0202	**3.5e-5**
Music JSB Chorales (loss)	300K	8.45	8.43	8.91	**8.10**
Music Nottingham (loss)	1M	3.29	3.46	4.05	**3.07**
Word-level PTB (perplexity$^\ell$)	13M	**78.93**	92.48	114.50	88.68
Word-level Wiki-103 (perplexity)	-	48.4	-	-	**45.19**
Word-level LAMBADA (perplexity)	-	4186	-	14725	**1279**
Char-level PTB (bpc$^\ell$)	3M	1.36	1.37	1.48	**1.31**
Char-level text8 (bpc)	5M	1.50	1.53	1.69	**1.45**

NOTE: the **number of hidden units** was chosen so that the model size is approximately at the same level as the recurrent models with which we are comparing.

TCN SETTINGS							
Dataset/Task	**Subtask**	k	n	**Hidden**	**Dropout**	**Grad Clip**	**Note**
The Adding Problem	$T = 200$	6	7	27	0.0	N/A	
	$T = 400$	7	7	27			
	$T = 600$	8	8	24			
Seq. MNIST	-	7	8	25	0.0	N/A	
		6	8	20			
Permuted MNIST	-	7	8	25	0.0	N/A	
		6	8	20			
Copy Memory Task	$T = 500$	6	9	10	0.05	1.0	RMSprop 5e-4
	$T = 1000$	8	8	10			
	$T = 2000$	8	9	10			
Music JSB Chorales	-	3	2	150	0.5	0.4	
Music Nottingham	-	6	4	150	0.2	0.4	
Word-level LM	PTB	3	4	600	0.5	0.4	Embed. size 600
	Wiki-103	3	5	1000	0.4		Embed. size 400
	LAMBADA	4	5	500			Embed. size 500
Char-level LM	PTB	3	3	450	0.1	0.15	Embed. size 100
	text8	2	5	520			

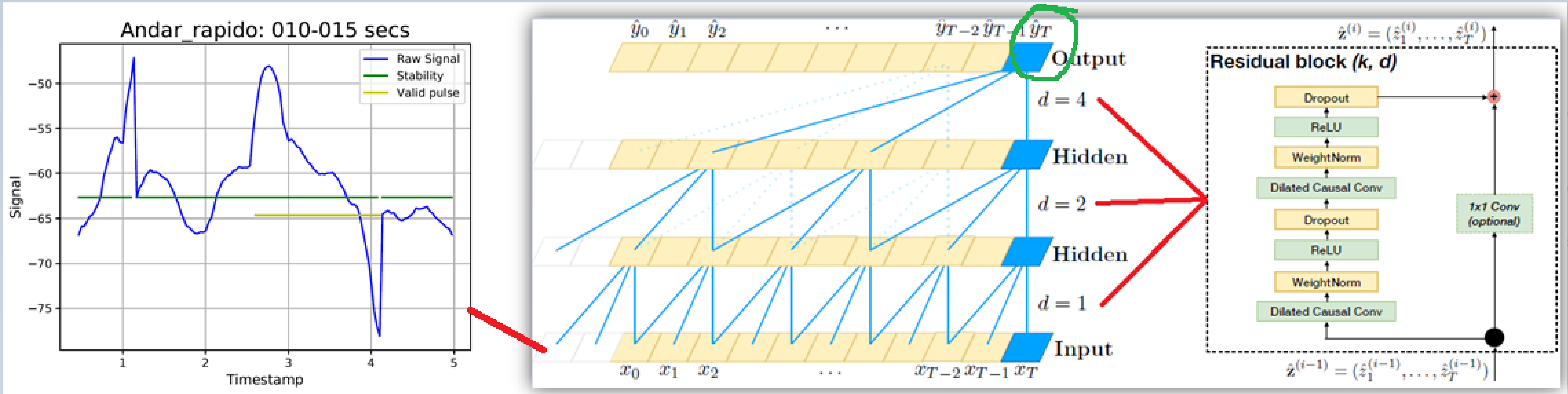
Sequence Modeling Task	Model Size ($\approx$)	Models			
		LSTM	GRU	RNN	**TCN**
Seq. MNIST (accuracy^h)	70K	87.2	96.2	21.5	**99.0**
Permuted MNIST (accuracy)	70K	85.7	87.3	25.3	**97.2**
Adding problem T=600 (loss$^\ell$)	70K	0.164	**5.3e-5**	0.177	**5.8e-5**
Copy memory T=1000 (loss)	16K	0.0204	0.0197	0.0202	**3.5e-5**
Music JSB Chorales (loss)	300K	8.45	8.43	8.91	**8.10**
Music Nottingham (loss)	1M	3.29	3.46	4.05	**3.07**
Word-level PTB (perplexity$^\ell$)	13M	**78.93**	92.48	114.50	88.68
Word-level Wiki-103 (perplexity)	-	48.4	-	-	**45.19**
Word-level LAMBADA (perplexity)	-	4186	-	14725	**1279**
Char-level PTB (bpc$^\ell$)	3M	1.36	1.37	1.48	**1.31**
Char-level text8 (bpc)	5M	1.50	1.53	1.69	**1.45**

1) “TCNs can be build to have very long effective history sizes, which means they have the ability to look very far into the past to make a prediction. To this end, a combination of very deep networks augmented with residual layers and dilated convolutions are deployed.”

2) “Generic TCN architecture outperform canonical recurrent architectures across a broad variety of sequence modelling tasks that are commonly used to benchmark the performance of recurrent architectures.”

Ruslan Padnevych n°47222
r.padnevych@campus.fct.unl.pt

MY THESIS



Andar_rapido: 010-015 secs
Raw Signal
Stability
Valid pulse
Signal
Timestamp
Output
Hidden
Hidden
Input
d = 4
d = 2
d = 1
Residual block (k, d)
Dropout
ReLU
WeightNorm
Dilated Causal Conv
Dropout
ReLU
WeightNorm
Dilated Causal Conv
1x1 Conv (optional)

Temporal Convolution Network - Liveness Detection

Hyperparameters

Kernel size k = 3

Dilation factor d = 2^*v*, with *v* the level of the network

Padding p = (k-1) * d