How to limit label dissipation in neural-network validation: Exploring label-free early-stopping heuristics

MAHYA AMERYAN and LAMBERT SCHOMAKER, University of Groningen, The Netherlands

In recent years, deep learning (DL) has achieved impressive successes in many application domains, including Handwritten-Text Recognition. However, DL methods demand a long training process and a huge amount of human-based labeled data. To address these issues, we explore several label-free heuristics for detecting the early-stopping point in training convolutional-neural networks: (1) Cumulative Distribution of the standard deviation of kernel weights (SKW); (2) the moving standard deviation of SKW, and (3) the standard deviation of the sum of weights over a window in the epoch series. We applied the proposed methods to the common RIMES and Bentham data sets as well as another highly challenging historical data set. In comparison with the usual stopping criterion which uses labels for validation, the label-free heuristics are at least 10 times faster per epoch when the same training set is used. The use of alternative stopping heuristics may require additional epochs, however, never requiring the original computing time. The character error rate (%) on the test set of the label-free heuristics is about a percentage point less in comparison to the usual stopping criterion. The label-free early-stopping methods have two benefits: They do not require a computationally intensive evaluation of a validation set per epoch and all labels can be used for training, specifically benefitting the underrepresented word or letter classes.

CCS Concepts: • **Computing Methodologies**; • **Artificial Intelligence**; • **Machine Learning**;

Additional Key Words and Phrases: deep learning, early-stopping criterion, convolutional neural networks, historical handwritten word recognition

1 INTRODUCTION

Today, various interesting applications, e.g., historical handwriting recognition, are performed using deep learning (DL) techniques [3, 21] due to ever-increasing classification accuracies. However, DL methods require a time-consuming training process and an abundance of human-labeled data for supervised learning. An important mechanism of modern neural-network training is *early stopping* of training of DL models, in order to prevent an overfit of models [6] on the given training set. This is achieved by bailing out of the training iterations at the point where the performance of the model is considered adequate. There exists a wide range of stopping criteria in the recent state of the art in deep learning [22], e.g., terminating the training process when the performance on a separate validation set is considered satisfactory. In other words, by using k-fold cross-validation, 10% to 20% of the annotated data is used to determine the moment of early stopping. This amount of labeled training data could have been employed for the primary task of training the required I/O transformation by the neural network. This problem manifests itself more when the data set is limited in terms of size. Especially in the case of historical manuscripts, there will be just a small amount of labels for a newly-discovered document in an unknown writing style. This raises the following research question:

We would like to thank Lutz Prechelt and two other anonymous reviewers for their useful and constructive comments on a previous version of this article.

Authors' address: Mahya Ameryan, m.ameryan@rug.nl/mahya.ameryan@gmail.com; Lambert Schomaker, L.R.B.Schomaker@rug.nl, University of Groningen, Nijenborgh 9, Groningen, Groningen, The Netherlands, 9747 AG.

XXXX-XXXX/2023/4-ART
https://doi.org/10.1145/3587168

'How is it possible to obtain reliable results from a trained neural network, without spilling valuable labels on the secondary process of validation?'. Rephrased, the question can also be read as: 'How to use all of the available labels for the primary training process?'. The urgency of label use is most strongly felt in the case of underrepresented classes. In textual data, the class imbalance is a given due to Zipf's Law [39], with many low-frequent words in what is called 'the long tail'. For characters, a similarly skewed distribution is present in many languages [10, 36].

The remainder of this paper is organized as follows. In Section 2, we briefly survey related studies on state-of-the-art techniques for early stopping. In Section 3, we explore several different early stopping approaches. The experimental evaluation and discussion are given in Sections 4 and 5, followed by a conclusion section.

2 RELATED WORK

Overfitting is a basic problem in supervised machine learning since it prevents the generalization of the models to unseen input patterns [13]. Overfitting occurs as the consequence of a mismatch between the number of free parameters (weights) and the limited size of the training set, given the complexity of a classifier, and the risk of stopping the training process at an inopportune iteration due to the eagerness of obtaining high classification accuracies [37].

The history of early-stopping heuristics dates back to the 1970s when research into the Landweber iteration was carried out [28]. In the 1990s, early stopping was used for neural networks [19] where the stochastic gradient descent is used to predict the performance of the model.

A common but very basic approach for early stopping uses only the training-loss values (e.g., L1 or L2 loss) where the training process stops after not observing a further decrease in the training-loss values after a predefined number of iterations. However, the training loss has fluctuated modification and it is difficult to predict the performance of a model on unseen data reliably. For this reason, independent validation data are used to decide whether training can stop. A common indicator of overfitting is an increase in validation loss while training loss still appears to be diminishing. Stopping on the basis of the validation loss delivers a better prediction of the performance on unseen data. However, a loss (or an accuracy) can only be computed if target values (i.e., labels) are known for each input/output sample pair. In order to be reliable, the number of validation samples needs to be large, which unfortunately is at the cost of available labeled data for the training process itself.

Table 1 shows character-error rates (CER) obtained in a 50-80 epoch range in a handwriting-recognition task, using 60 or 80 percent of the labeled data for the training, proper. The data set contains 9,630 word images [1]. The data set contain 52 different characters. The result is derived from a neural network containing 12 convolutional layers [17] followed by a CTC layer [8] containing 18,764,416 trainable parameters. The number of outputs are 54 (52 characters + an special space for CTC [8] + a stressed-ending token [2, 3]). It can be seen that the CER drops 0.4 to 2.9 percentage points if an extra 20% of labeled training data is used, which corresponds to an error reduction (relative gain) of 3-18 percent. Given the current focus on fractional improvements in deep learning performance, this difference is not negligible.

Given such a potential for error reduction provided by the availability of enough labeled samples, it becomes interesting to consider methods that are able to decide for an early stop of the training without the label spilling that is associated with the usual validation procedure. A challenging aspect concerns the measurement of performance. In the application itself, loss values are only indirect performance indicators. In handwriting recognition, classification accuracy is the final, relevant indicator of network performance. Such accuracy can only be computed using the known labels, for samples in the validation set. If a new early-stopping criterion is introduced, one would like to know the expected validation and test accuracies at the point of bailing out [12]. That such criteria may exist, can be seen in Fig. 7, as a preview.

Table 1. Character error rates (Avg ±sd) obtained in the training range of 50-80 epochs in a handwriting-recognition task, using 60 vs 80 percent of the MkS data set [1] for the training. It can be seen that the availability of yet another portion of 20% of the training data, e.g., by preventing label dissipation in the validation, is attractive. (pp = percentage point)

Portion of labeled data set	training set size	Epoch			
		50	60	70	80
60%	5778	15.4 ±1.3	16.0 ±1.8	13.2 ±0.8	13.2 ±1.1
80%	7704	14.5 ±0.9	13.1 ±1.5	12.5 ±0.7	12.8 ±0.7
Gain (pp)	-	0.9	2.9	0.7	0.4
Relative Gain (%)	-	5.84	18.13	5.30	3.03

In [22], it is proposed that a careful selection of a stopping criterion requires a consideration of the following questions: (a) Training time: How much each criterion affects the training time?; (b) Efficiency: How much of the training time is superfluous using each criterion?; (c) Effectiveness: How does each criterion determine the final performance?; (d) Robustness: How robust are mentioned qualities with respect to the random initial condition or network architecture?; (e) Trade-offs: Which criterion offers the best time-performance trade-off?; (f) Quantification: How to measure the quantity of the trade-off?

In [22], it is commemorated that early work in neural networks focused primarily on training performance and differences between stopping heuristics were not considered important [31, 35]. In more recent years, however, reliable performance prediction has become an important issue. In the NIPS performance prediction and model selection challenges, the focus was rightly on an accurate prediction of model performance rather than on high training performances per se. In these benchmarks, the winner is not the team with the highest, cherry-picked performance, but the team which most accurately predicted the performance on completely unseen data [12].

In literature, different properties of a neural network are considered in order to prevent overfit and/or to optimize performance, e.g., by looking at the evolution of weight values [4, 22]. A graph-theoretical method, based on algebraic topology using network weights and connectivity, is proposed in [25] for early stopping. In [29], in order to optimize a Multi-Layer Perceptron (MLP), a method based on monitoring the entropy of all weight values in the model is proposed, based on (expensive) particle-swarm optimization. In the current study, we will also focus on the properties of the set of weights in a model, with the aim of finding a simple and effective method that can be used for different network architectures. The idea is that starting at the first epoch with uniformly distributed weight values that break the symmetry, the distribution will change over time as (local) solutions have been found. The goals of the current study are:

- To use as much labeled data for training, per se, and not dissipate it for the secondary task of validation (i.e., to deal with label sparsity);
- To monitor the statistics of the weight values, inspired by [4, 22];
- To find a stopping epoch that does not differ significantly from what a regular validation-loss-based criterion finds;
- To identify a stopping epoch that has a character error rate that does not differ much from what a traditional validation-based method would find;
- To use a measure that fluctuates less drastically than the loss which is typically a ragged curve [30, 38].

3 METHOD

In this section, we explore three early-stopping heuristics for the training of convolutional neural networks (CNNs [17]). Convolutional neural networks avoid unwieldy and computationally expensive processing of full images by using a battery of compact, trainable image filters which use only a fraction of the coefficients that would have been needed in the case of traditional neural networks, where hidden units in the first layer need to be connected to the whole image. The information contained in the values of the weights of all kernels is an essential determinant of the success of a convolutional neural network. The training of such networks requires a validation stage, where labeled data is needed, reducing the effective size of the training set, proper. In this section we will introduce three heuristics for early stopping as an alternative approach to the usual validation. The proposed heuristics are: 1 Cumulative Distribution of the Standard deviation of Kernel Weights (CD-SKW); 2 Moving Standard deviation of SKW (SSKW); 3 Moving Standard deviation of the Total sum of Kernel Weights (STKW). By monitoring the changes in the value of a defined measure over the training process, we apply an approach for the determination of a stopping point which is detailed in this section. The three approaches are evaluated using several neural-network architectures, which will be explained later.

3.1 Introducing SKW: Using the standard deviation of kernel weights

Here, we define a new criterion, the standard deviation of kernel weights (SKW) as it evolves during training. To compute SKW, we compute the standard deviation of the CNN kernel weight values as obtained at the end of each epoch t, yielding a series SKW_t representing the evolution of weight-value variation over the epochs. We have found in pilot experiments that the SKW_t curve appears to be increasing and is relatively smooth. Therefore, it was hypothesized that this measure may lend itself as an early stopping criterion, if a sensitive detection can be made of a statistical change in its evolution. Whereas the continuous increase may appear as a non-stationary process, the growth itself appears to be regular. Therefore, a non-stationarity test on growth can be used to detect a change in the process of weight changes[18, 23, 24]. The function *compute_SKW* in Algorithm 1 in the Appendices shows the details. The list of symbols is as described in Table 10.

3.1.1 Heuristic 1: Early stopping with Cumulative distribution of SKW (CD-SKW).

As stated earlier, the two commonly used early-stopping methods in deep learning are based on the loss value of the training set ($Loss(train)$[1]) or the character error rate of the model on the validation set (CER_{val}). Both of these traditional measures gradually converge to an asymptote at some interval of epochs during the training process. However, the SKW value surprisingly continuously increases during the training process as we will show further in Section 4.3. To deal with this issue, we need to detect a statistically unexpected growth value for SKW to be defined as a stopping point. By looking at a running SKW over a window of ω epochs, the local statistics of SKW can be estimated and the probability of a new incoming SKW value x can be determined. Various methods can be used for this goal. We have experimented with some methods [9, 16], however, with limited results. Finally, a basic statistical test appeared to yield satisfying results. We consider a lower bound on the probability ($p(x)$) of the SKW criterion occurring assuming a Gaussian distribution of the SKW from the previous ω epochs. The value $p(x)$ is calculated from the Cumulative distribution function (CDF). When p is smaller than a predefined threshold, the new SKW value is unlikely to be generated by the same statistical process, and training is terminated. We call this the Cumulative distribution of the standard deviation of kernel weights (CD-SKW) early-stopping heuristic. In the *main* function of the Algorithm 1, the CD-SKW early-stopping method is applied.

Assuming that SKW values have been collected over a time window of ω epochs, we can estimate a Gaussian model for the time series, returning $p(x)$ for a new SKW value x:

$$\mathcal{N}_\omega(x, \mu_\omega, \sigma_\omega) \;= 1/(\sigma_\omega\sqrt{2\pi})exp(((x-\mu_\omega)/(2\sigma_\omega))^2) \tag{1}$$

[1]In the sequel we will use the generic word 'loss', referring to the actually used CTC loss for LSTMs in HTR

where μ_ω and σ_ω are the running average and running standard deviation. In order to obtain a probability estimate suitable for thresholding, we need to integrate to obtain the cumulative distribution:

$$CDF_\omega(x, \mu_\omega, \sigma_\omega) = \int_{-\infty}^{x} \mathcal{N}_\omega(v, \mu_\omega, \sigma\omega)dv \quad (2)$$

Finally, μ_ω, the moving average of SKW and σ_ω, the moving standard deviation of SKW over the previous ω epochs are computed as:

$$\mu_\omega = (\sum_{y=t-\omega}^{t} SKW_y)/\omega \quad (3)$$

$$\sigma_\omega = \sqrt{\sum_{y=t-\omega}^{t} (SKW_y - \mu_\omega)^2)/\omega} \quad (4)$$

3.1.2 *Heuristic 2: Early stopping with moving standard deviation of SKW (SSKW).*

In heuristic 2, we use another approach for monitoring the evolution of SKW. As discussed before, the SKW values continuously increase over the training process and do not converge to an asymptote. However, our pilot experiments show that the moving standard deviation of SKW decreases after the first run-in stage of the neural network. Therefore, we use the moving standard deviation of SKW over a window sized ω_1 for detecting the stopping point. We call this approach the early-stopping method based on the moving standard deviation of SKW ($SSKW$). After observing no decrease in $SSKW$ for ω_2 epochs, we terminate the training process.

3.2 Heuristic 3: Early stopping using the moving standard deviation of the total sum of the kernel weights (STKW)

In this approach, we define another criterion, the evolution of the total sum of the kernel weights (TKW) during training. The TKW is a scalar value and is calculated by summing all the kernel weights of the CNN model together at the end of each epoch t, leading to another series TKW_t. The function $compute\_TKW$ in Algorithm 2 in Appendices shows the details. The list of symbols is shown in Table 10.

Our pilot experiments show that the TKW values are decreasing rather smoothly in the training process after the first run-in stage of the neural network.

However, like the other two heuristics, the curve does not converge to an asymptote during the training process and continues to decrease. Therefore, we do not consider the growth of the TKW_t itself, but calculate the running standard deviation over a window of ω epochs ($STKW_t$). The evolution of $STKW_t$ shows a great similarity with the evolution of the character error rate of the model on the validation set (CER_{val}), as will be shown later.

After preliminary experiments, it appeared that three different window sizes ω need to be taken into account. First, we do not want to stop in the first few ω_1 epochs (run-in stage), because the network is settling from the potentially far-away random initialization. Then, after ω_1 epochs, we start to calculate the running standard deviation of TKW_t over ω_2 epochs at each step. Lastly, we check if the minimum of $STKW_t$ does not decrease for ω_3 epochs and stop at epoch t of this minimum. For our experiments we set all three window sizes to the same value, that is $\omega = \omega_1 = \omega_2 = \omega_3$. The moving standard deviation of TKW over the previous ω epochs (γ_ω) are calculated as:

$$\gamma_{\omega_2} = \sqrt{\sum_{y=t-\omega_2}^{t} (TKW_y - \mu_{\omega_2})^2)/\omega_2} \quad (5)$$

where the μ_{ω_2} is the moving average of ω_2 previous TKW values:

$$\mu_{\omega_2} = \Big(\sum_{y=t-\omega_2}^{t} TKW_y \Big) / \omega_2 \tag{6}$$

3.3 Convolutional neural networks

We apply the three early-stopping heuristics to several convolutional neural network architectures. First, an input image is preprocessed as detailed in [1]. Then, it is processed by CNN layers. Each layer of a CNN has a convolution operation, normalization method, the ReLU activation function [20], and a max-pooling procedure. The initialization method (initializing distribution), learning rate, input image size, the hidden unit sizes, and the number of trainable parameters are shown in Table 2. For Heuristic 1, the learning rate is scheduled and is 0.003 until epoch 25. At epoch 26 it drops to 0.002. From epoch 70, the learning rate is 0.001. For Heuristic 2 and 3, the learning rate is constant.

For all the architectures, the kernel filters have the size of 3×3. The Adam optimizer is used [15]. No dropout is used in the models. The last layer of CNN is followed by a Connectionist Temporal Classification (CTC [8]) and the dictionary-free Best-Path method [7] is used for decoding.

3.4 Data set

In our experiments, we use three data sets: MkS [1] for evaluating Heuristic 1, and RIMES [11] and Bentham [5, 27] for evaluating Heuristic 2 and 3. The MkS data set is collected from the extensive biological field notes of the Natuurkundige Commissie (NC) in the Indonesian Archipelago in the period 1820-1850 [32–34]. The original manuscript contains 17,000 pages and is preserved in the Naturalis Biodiversity Center in Leiden, the Netherlands. Variant languages are used in the NC manuscript, including German, Dutch, Latin, Malay, French, and Greek. Scatteredly, 950 pages of the NC corpus are annotated for this project (NNM001001033-7). We conducted the 5-fold cross-validation after random data splitting. The proportion of words in the test set not appearing in the training set, out-of-vocabulary (OOV) words, is 29.5% in a case-insensitive manner and 31.9% in a case-sensitive

Table 2. Configuration of three convolutional neural network architectures (Arch.: A_i, $i = 1, ..., 19$) used in the evaluations the Heuristic 1, 2, and 3. Please refer to the text for further explanation.

Heuristic	Arch.	initializing distribution	Learning rate	Input image size	#hidden units	hidden units size	#trainable parameters
Heuristic 1	A_1	truncated normal	Scheduled	128×32	12	128(2), 256(2), 512(8)	18,764,416
	A_2	truncated normal	Scheduled	128×32	14	128(2), 256(3), 512(9)	21,715,072
	A_3	truncated normal	Scheduled	128×32	16	128(2), 256(3), 512(11)	26,435,712
Heuristic 2	A_4	truncated normal	0.001	128×32	15	128(1), 256(2), 512(12)	28,073,856
	A_5	normal	0.001	128×32	18	128(2), 256(4), 512(12)	29,402,240
	A_6	truncated normal	0.001	128×32	18	128(2), 256(4), 512(12)	29,402,240
	A_7	truncated normal	0.002	128×32	18	128(2), 256(4), 512(12)	29,402,240
	A_8	Uniform	0.001	128×32	18	128(2), 256(4), 512(12)	29,402,240
	A_9	truncated normal	0.001	1200×64	13	128(3), 256(7), 512(3)	10,083,712
	A_{10}	truncated normal	0.002	1200×64	13	128(3), 256(7), 512(3)	10,083,712
	A_{11}	truncated normal	0.003	1200×64	13	128(3), 256(7), 512(3)	10,083,712
	A_{12}	truncated normal	0.001	1200×64	15	128(3), 256(8), 512(4)	13,034,368
	A_{13}	truncated normal	0.001	1200×64	19	128(4), 256(7), 512(8)	22,033,024
Heuristic 3	A_{14}	Uniform	0.001	128×32	15	128(1), 256(2), 512(12)	28,073,856
	A_{15}	Uniform	0.001	128×32	18	128(2), 256(4), 512(12)	29,402,240
	A_{16}	Uniform	0.001	128×32	17	128(5), 256(3), 512(9)	22,174,080
	A_{17}	Uniform	0.001	1200×64	14	128(3), 256(7), 512(4)	12,444,032
	A_{18}	Uniform	0.001	1200×64	15	128(3), 256(8), 512(4)	13,034,368
	A_{19}	Uniform	0.001	1200×64	18	128(4), 256(6), 512(8)	15,542,400

manner. Table 3 shows the summarization of the MkS data set along with its subsets in the terms of the number of image samples, and the lexicon size in the case-(in)sensitive manner.

RIMES is a contemporary public benchmark data set for handwriting recognition. We use the isolated words of ICDAR 2011. The historical Bentham data set is a publicly available benchmark and is derived from the Bentham archive. Bentham is annotated on line level and partially on word level. This manuscript was written by two authors, English philosopher and reformer Jeremy Bentham (1748-1832) and his secretarial staff. The text is principally in the English language, and partly in Greek and French. We use the stressed-ending labeling-coding scheme [2, 3] for the experiments on MkS and RIMES. In this method, an extra character, here the bar sign (|), is added to the ground-truth labels.

4 RESULTS

In this section, we first describe how our experiments are conducted. After that, the numerical results of the three heuristics are presented.

When Heuristic 1, 2, and 3, and also the early-stopping methods based on $Loss(train)$ are used, the CER_{val} and CER_{test} are computed after the training process. The neural network architectures are implemented on the TensorFlow framework.

Table 3. The MkS data set. Subscript *cs* and *ci* denote case sensitive and case insensitive for counting the lexicon size [1].

Set		#Image samples	$\#Lexicon_{cs}$	$\#Lexicon_{ci}$
MkS	Train	5,778	2,648	2,497
	Validation	1,926	1,188	1,142
	Test	1,926	1,188	1,142
	Total	9,630	3,730	3,494

Table 4. Result of the convolutional neural networks $A_i = A_1, ..., A_3$ in Table2 on the MkS data set using the early-stopping heuristic based on the character error rate (CER) of the validation set, the loss of model on the training set, and label-free CD-SKW early-stopping (heuristic 1).

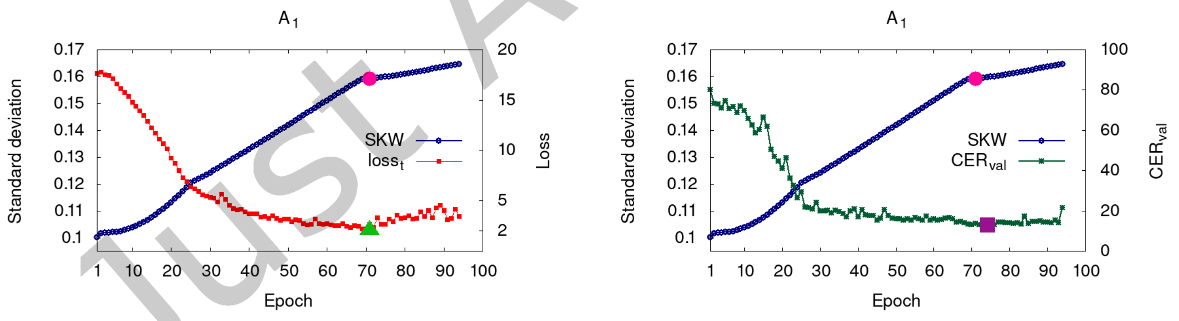
early-stopping method	Arch.	Stopping epoch	Training loss	CER_{val}	CER_{test}
Loss	A_1	70.6 ±0.5	1.8±0.2	12.8 ±0.4	13.1 ±0.2
	A_2	69.4 ±1.6	2.2 ±0.2	13.1 ±1.5	13.4 ±1.5
	A_3	69.4 ±3.3	2.2 ±0.5	13.9 ±1.8	13.7 ±1.9
	Avg ±sd	69.8 ±2.2	2.1 ±0.4	13.3 ±1.4	13.4 ±1.4
CER_{val}	A_1	72.2 ±11.8	2.7 ±0.5	12.7 ±0.8	13.5 ±1.2
	A_2	72.0 ±11.4	2.8 ±0.6	12.7 ±0.7	13.1 ±0.6
	A_3	73.6 ±1.6	2.9 ±0.3	11.5 ±0.5	12.5 ±0.4
	Avg ±sd	72.6 ±9.6	2.8 ±0.5	12.3 ±0.9	13.0 ±0.9
CD-SKW	A_1	69.4 ±1.6	2.4 ±0.2	13.1 ±1.5	13.5 ±1.5
	A_2	69.4 ±1.6	2.4 ±0.2	13.1 ±1.5	13.5 ±1.5
	A_3	69.4 ±1.6	2.4 ±0.2	13.1 ±1.5	13.5 ±1.5
	Avg ±sd	69.4 ±1.6	2.4 ±0.2	13.1 ±1.5	13.5 ±1.5

4.1 Quantitative results of Heuristic 1, CD-SKW

Table 4 shows the comparison of two well-known early-stopping approaches based on character error recognition (CER(%)) on the validation set and the training loss with the suggested label-free CD-SKW early stopping heuristic in terms of average and standard deviation ($Avg \pm sd$) of the stopping epoch, $LOSS(train)$, the CER of the model on the validation (CER_{val}), and test (CER_{test}) sets. The used data is the MkS data set. Each row of $A_i,\ i = 1, 2, 3$ shows the average and standard deviation of five trained networks (one architecture on five rounds of 5-fold cross-validation). The row at the bottom of $A_i,\ i = 1, 2, 3$ shows the result of the 15 trained models (3 architecture × five rounds). To detect the stopping point by applying the label-free CD-SKW early-stopping heuristic, no labeled data is used. In the label-free CD-SKW early-stopping heuristic, we consider the interval size for investigating the occurrence of an early stopping ($\omega = 15$) and the threshold of the probability defined in Algorithm 1 ($\tau = 0.90$) in the Appendices. For the comparison with the two other stopping methods, i.e., one based on the CER on the validation set and one on the training loss, we use the same window size as in the SKW approach, i.e., selecting an epoch when the model did not improve over the last 15 epochs.

When the early-stopping method based on loss and also CD-SKW is used, the CER_{val} and CER_{test} sets are computed after the training process. As reported in Table 4, when CD-SKW is used, on average CER_{val}=13.1 is 0.2 pp less than when the training loss is used (CER_{val}=13.3). When the training loss is used, CER_{test} is 0.1 pp less than when CD-SKW is used. It reveals that the two methods yield highly similar results, although SKW is not using label information. For practical use, the differences between the results are small enough (< 0.5 pp on CER_{test}) to develop some trust in the label-free CD-SKW heuristic.

Fig. 1a shows the evolution of the SKW value (the blue line with circles) in comparison with $LOSS(train)$ (the red line with squares) and CER_{val} (the green line with crosses) in Fig. 1b during the training process. the values are conducted using the architecture A_1 in Table 2. In this example, the network was trained on one training set, of one round of 5-fold cross-validation of the MkS data set [1, 34]. Results for other rounds were similar (cf. the standard deviations in Table 4). In Fig. 1a and 1b, the pink dot on the SKW line indicates the epoch=71 where the label-free SKW early-stopping method recognizes the proper epoch for terminating the training. That



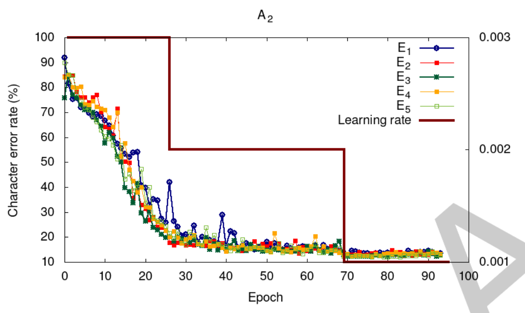
(a) SKW vs Loss (b) SKW vs CER

Fig. 1. Comparison of the evolution of the SKW curve with $train(LOSS)$, and CER_{val} curves: (a) The increasing blue line with circles shows the SKW curve, where the pink circle indicates the stopping point determined by the CD-SKW method detecting the discontinuity. The decreasing red line with square points illustrates the loss values of the model on the training set, where the green triangle presents the stopping point determined by the training loss-based early-stopping approach. (b) The figure shows the Comparison of the SKW curve with the decreasing green curve of the character error recognition of the validation curve (CER_{val}). The values are obtained from applying the architecture A_1 in Table 2 on one round of 5-fold cross-validation of the training set of MkS data set (60% of the MkS data set). No validation set is used.

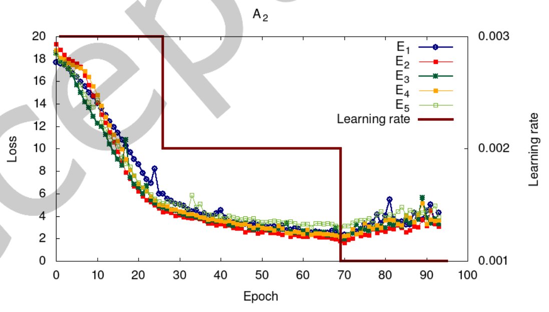
is, the probability of the SKW given the previous 15 epochs ($\omega = 15$) is smaller than the predefined threshold ($\tau = 0.90$). In this figure, the method based on the loss value found the same epoch. However, there will generally be slight differences in the point of stopping. As shown in Table 4, the stopping epoch of A_1 based on the training loss equals 70.6 ±0.5 and based on SKW equals 69.4 ±1.6. After the pink dot and the green triangle the training process should be terminated. We continued training to show the behavior of the criteria. In Fig. 1b, the green line with crosses illustrates CER_{val}. The square point shows the stopping epoch. The SKW values, unlike the $LOSS(train)$ and CER_{val}, increase during the training process. Moreover, SKW has a clear increasing behavior while the $LOSS(train)$ and CER_{val} change with fluctuation.

Fig. 2 shows the effect of the learning rate on the character error rate (%) on the validation set, training loss, and the suggested CD-SKW in the A_2 architecture in Table 2 on five rounds of 5-fold cross-validation ($E_j,\ j = 1, \ldots, 5$).

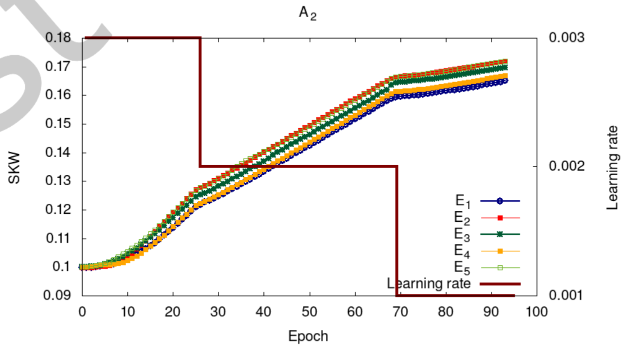
The computing time of CER_{val} and CD-SKW are shown in Table 5 for the neural networks (Arch.) in Table 2. The numerical values are in seconds per epoch. The loading (I/O) time of the validation set (29 seconds) is not considered in the computing of CER_{val}. Please note a considerable time gain factor of 10 to 20.



(a) The character error recognition



(b) The training (CTC) loss



(c) The standard deviation of the kernel weights

Fig. 2. The effect of the learning rate on the character error rate (%) on the validation set, training loss, and the suggested SKW in the A_2 architecture in Table 2 on five rounds of 5-fold cross-validation ($E_j,\ j = 1, \ldots, 5$).

Table 5. The comparison between the computing time of the character error rate of the validation set (CER_{val}) and the CD-SKW early-stopping method of the architectures in Table 2 for each epoch. The values are in seconds per epoch. The loading (I/O) time of the validation set is 29 seconds. This time is not regarded in computing of CER_{val}. We used NVIDIA V100 and 2.52G RAM.

Arch.	CER_{val}	CD-SKW
A_1	7.9	0.4
A_2	4.9	0.4
A_3	10.0	0.6

Fig. 3. Evolution of the normalized SKW variable during training, for CNN models A_4 to A_8 in Table 2 on the RIMES data set.

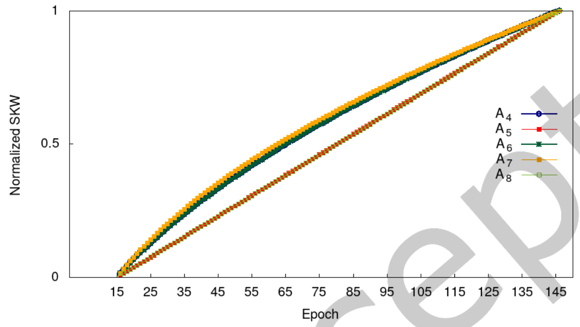
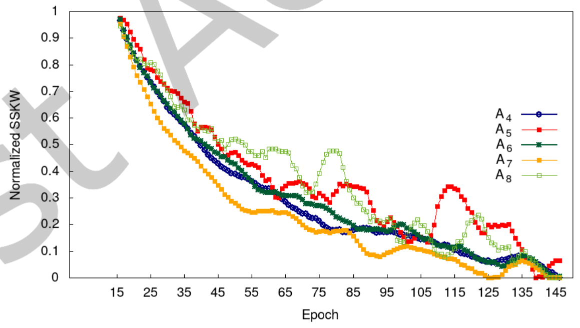


Fig. 4. Evolution of the normalized SSKW variable during training, for CNN models A_4 to A_8 in Table 2 on the RIMES data set.



4.2 Quantitative results of Heuristic 2, SSKW

The heuristic 2, $SSKW$, is used for the CNN architectures $A_i, i = 4, ..., 13$ in Table 2 which have a constant learning rate throughout the training. The architectures are applied to two publicly available benchmark data sets, RIMES and Bentham. The five architectures $A_i, i = 4, ..., 8$ are applied to the RIMES data set and the five architectures $A_i, i = 9, ..., 13$ are applied to the Bentham data set. By using a constant learning rate, the curve of the evolution of the SKW values during the training process does not show an (artefactual) bending point. The SKW values increase without converging to an asymptotic line. Fig. 3 shows the evolution of SKW of architectures $A_i, i = 4, ..., 8$ on RIMES. For better visibility, the values of the figures are normalized. Fig. 4 shows the evaluation of the normalized moving standard deviation of SKW ($SSKW$) values during the training for CNN models A_4 to A_8 in Table 2 on the RIMES data set. We observe that the $SSKW$ values decrease during the training process. The window size for the moving standard deviation is 15 epochs.

Fig. 5 shows the evolution of the normalized SSKW and CER_{val} values during training, for the architecture A_7 in Table 2 on the RIMES data set. The SSKW and CER_{val} show similarities in their evolution manner. The Pearson correlation is 0.90. The initial part of epochs < 15 is ignored to allow the CNN model to settle from the initial randomization, also letting the windowed computations to 'run in'.

Table 6 shows the comparison of $SSKW$ with two traditional early-stopping approaches based on CER_{val} and $Loss(train)$ in terms of average and standard deviation ($Avg \pm sd$) of stopping epoch, $Loss(train)$, $CER_{val}(\%)$

Fig. 5. Evolution of the normalized SSKW variable and the normalized CER_{val} values during training, for CNN model A_7 in Table 2 on the RIMES data set. The absolute values of CER_{val} are shown in Table 6. The Pearson correlation is 0.90. The initial part of epochs < 15 is ignored because of the 'run-in' stage of the neural-network training.

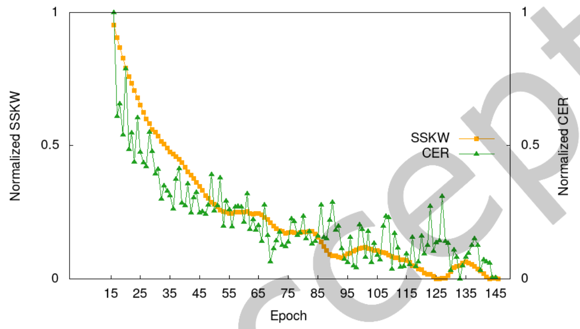


Table 6. Comparision of Heuristic 2, $SSKW$, with the early-stopping methods based on CER_{val} and $LOSS(train)$. Please refer to the text for further explanation.

Data set	ω_2	ω_1	Early-stopping method	epoch	Loss	CER_{val}	CER_{test}
RIMES	15		$Loss(train)$	>200	-	-	-
			CER_{val}	78.80 ±12.50	0.14 ±0.03	4.71 ±0.96	4.68 ±0.92
		15	$SSKW$	136.60 ±22.52	0.08 ±0.03	4.55 ±0.91	4.49 ±0.87
		10	$SSKW$	99.40 ±16.86	0.11 ±0.06	4.79 ±0.99	4.77 ±1.01
	17		$Loss(train)$	>200	-	-	-
			CER_{val}	90.60 ±13.15	0.12 ±0.03	4.59 ±0.90	4.60 ±0.88
		15	$SSKW$	140.00 ±26.69	0.08 ±0.03	4.56 ±0.89	4.50 ±0.87
		10	$SSKW$	110.60 ±29.06	0.11 ±0.06	4.76 ±1.03	4.76 ±1.02
Bentham	15		$Loss(train)$	>200	-	-	-
			CER_{val}	86.20 ±9.62	8.10 ±0.57	4.56 ±0.20	4.62 ±0.28
		15	SSKW	120.40 ±22.77	7.01 ±0.98	4.65 ±0.26	4.62 ±0.28
		10	$SSKW$	97.60 ±20.71	7.61 ±0.82	4.83 ±0.29	4.70 ±0.29
	17		$Loss(train)$	>200	-	-	-
			CER_{val}	89.60 ±15.47	7.97 ±0.76	4.54 ±0.23	4.60 ±0.29
		15	$SSKW$	124.40 ±17.07	6.99 ±0.95	4.67 ±0.28	4.67 ±0.33
		10	$SSKW$	107.20 ±18.78	7.50 ±0.68	4.75 ±0.17	4.68 ±0.25

Table 7. The comparison between the computing time of the early-stopping based on CER_{val} and the SSKW early-stopping method of the architectures A_4 to A_8 in Table 2 on RIMES for each epoch. The values are in seconds per epoch. The loading (I/O) time of the validation set is 40 seconds. This time is not regarded in this table. We used NVIDIA V100.

Arch.	CER_{val}	SSKW
A_4	37.7	0.5
A_5	39.1	0.6
A_6	38.7	0.6
A_7	38.3	0.6
A_8	39.0	0.6

and $CER_{test}(\%)$ of the five architectures $A_i, i = 9, ..., 13$ on the Bentham data set. The size of the window for computing $SSKW$ is $\omega_1 = 15\ or\ 10$. The training stops using the aimed criterion when it does not decrease for $\omega_2 = 15\ or\ 17$ epochs. The maximum number of training epochs equals 200. In our experiments, when the $Loss(train)$ early-stopping method is considered, at least one network needs more than 200 epochs. Therefore, training is terminated without using an early stopping method. We indicate this with $'-'$ in Table 6.

Table 6 shows that when $SSKW$ is used for RIMES and $\omega_1 = 15$, $CER_{test}(\%)$ is less than when CER_{val} is used as criterion ($\omega_2 = 15$: 4,49 vs 4,68, $\omega_2 = 17$: 4,50 vs 4,60). When $\omega_1 = 10$, $CER_{test}(\%)$ obtained by SSKW is more than when CER_{val} is used as criterion ($\omega_2 = 15$: 4,77 vs 4,68, $\omega_2 = 17$: 4,76 vs 4,60). $SSKW$ needs 20 to 58 epochs more compared to the early-stoping method based on CER_{val}. Moreover, $SSKW$ demands shorter training compared to the early-stoping method based on $Loss(train)$.

Table 6 shows that when $SSKW$ is used for Bentham and $\omega_1 = 15$, $CER_{test}(\%)$ equals to when CER_{val} is used as criterion or less than 1 pp higher ($\omega_2 = 15$: 4.6 vs 4.6, $\omega_2 = 17$: 4.67 vs 4.6). When $\omega_1 = 10$ and $SSKW$ is considered as the criterion, $CER_{test}(\%)$ is 0.08 pp more than when CER_{val} is used as criterion ($\omega_2 = 15$: 4.70 vs 4.62, $\omega_2 = 17$: 4.68 vs 4.60). $SSKW$ needs 11 to 35 epochs more compared to the early-stoping method based on CER_{val}. Also, $SSKW$ takes shorter training in comparison with the early-stoping method based on $Loss(train)$.

The computing time of the early-stopping method based on CER_{val} and the $SSKW$ early-stopping method are shown in Table 7 for the architectures (Arch.) A_4 to A_8 in Table 2. The numerical values are in seconds per epoch. The loading (I/O) time of the validation set (40 seconds) is not considered in computing of CER_{val}. Please note a considerable time gain factor of 60 to 75.

4.3 Quantitative results of Heuristic 3, STKW

The evolution of the normalized TKW during training of the CNN models A_{14} to A_{19} in Table 2 on RIMES and Bentham is shown Fig. 6. Like, the SKW values, the TKW values grow without converging to an asymptotic line. Fig. 6 shows that the curve does not have a bending point. However, the evolution of $STKW$ and CER_{val} are highly similar (Pearson correlation: r=0.89). Fig. 7 shows the evolution of the normalized values of $STKW$ and CER_{val} of A_{14} applied to the RIMES data set. The initial part of the training process (epochs < 30) is disregarded to let the model settle from the random initialization.

To evaluate the heuristic 3, $STKW$, we applied the architectures $A_i, i = 14, 15, 16$ in Table 2 on the RIMES data set, and the architectures $A_i, i = 17, 18, 19$ on Bentham. The comparison of the $STKW$ with two early-stopping approaches based on CER_{val} and $Loss(train)$ in terms of average and standard deviation ($Avg\ \pm sd$) of stopping epoch, $Loss(train)$, $CER_{val}(\%)$ and $CER_{test}(\%)$ of the models is reported in Table 8. For the early-stopping method based on $Loss(train)$, CER_{val}, and $STKW$, ω in Table 8 shows the number of epochs that the model does not show improvement as well as the other two windows of $STKW$, i.e., the window which shows the run-in stage

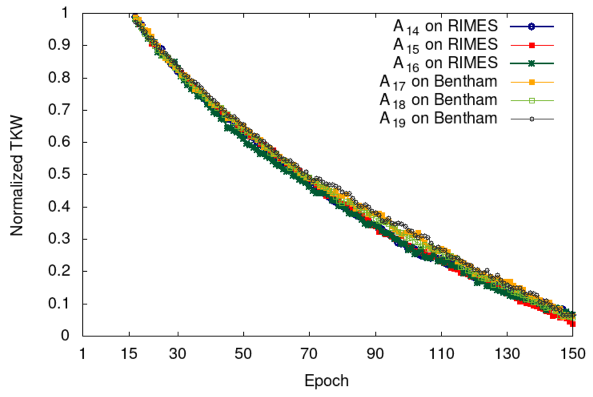


Fig. 6. Evolution of the normalized TKW variable during training, for CNN models A_4 to A_9 in Table 2 and for two data sets.

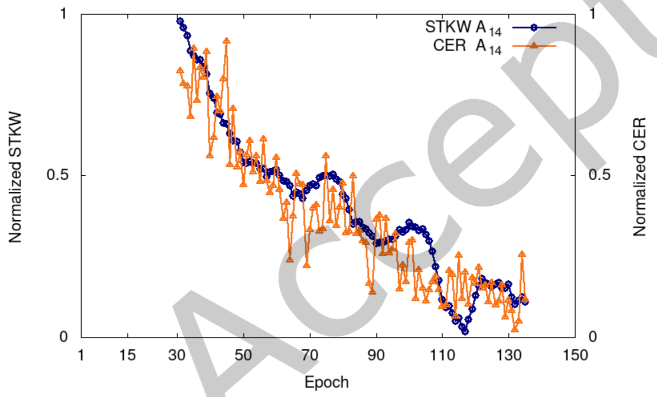


Fig. 7. The normalized CER_{val} (orange with rectangles) *vs* the normalized STKW values (blue with circles) of the A_{14} CNN model in Table 2 which also shows the absolute CER values. The Pearson correlation is 0.89. The initial part of epochs < 30 is ignored to allow the network to settle from the initial randomization, also letting the windowed computations to 'run in'.

and the window for the standard deviation of TKW_t. In the evaluations, we consider the maximum number of the training epoch equal to 200.

For RIMES, when $Loss(train)$ is the stopping criterion, the stopping point is not detected during the 200 training epochs. We show this observation with $'-'$. For RIMES, for both $\omega = 15$ and $\omega = 17$, the average of CER_{test} is equal when the STKW method is used. The size of ω does not show an effect on CER_{test} on average. For RIMES, in comparison with the early stopping method based on CER_{val}, the $STKW$ detects the stopping point 5 to 7 epochs later with 0.25 to 0.34 pp higher CER_{test}. For $STKW$, the higher ω size shows less stability.

For Bentham, the $STKW$ method detects the stopping epoch 10 to 29 epochs before CER_{val} and about 55 epochs before the early-stopping method based on $Loss(train)$. The CER_{test} obtained by $STKW$ on Bentham is 0.6 to 0.9 pp more than the CER_{test} obtained by the early-stopping method based on CER_{val}. The CER_{val} and CER_{test} derived by the early-stopping method based on $Loss(train)$ are up to 0.8 pp less than CER_{val} and CER_{test} derived by the early-stopping method based on STKW.

Table 8. Comparision of Heuristic 3, $STKW$, with the early-stoping methods based on CER_{val} and $LOSS(train)$. Please refer to the text for further explanation.

Data set	ω	Early-stopping method	epoch	Loss	CER_{val}	CER_{test}
RIMES	15	$Loss(train)$	>200	-	-	-
		CER_{val}	90.33 ±20.8	21.38 ±1.7	6.15 ±0.1	6.16 ±0.2
		STKW	97.00 ±15.5	19.39 ±2.2	6.40 ±0.4	6.41 ±0.4
	17	$Loss(train)$	>200	-	-	-
		CER_{val}	99.00 ±22.1	20.09 ±0.4	6.02 ±0.2	6.07 ±0.3
		STKW	104.33 ±6.1	19.83 ±1.2	6.43 ±0.5	6.41 ±0.6
Bentham	15	Loss(train)	127.33 ±5.4	9.46 ±0.59	6.85 ±0.18	6.91 ±0.11
		CER_{val}	83.00 ±15.90	153.44 ±11.80	6.80 ±0.13	6.86 ±0.15
		STKW	72.67 ±5.19	153.98 ±11.09	7.54 ±0.14	7.49 ±0.08
	17	Loss(train)	129.00 ±6.0	9.16 ±0.5	6.93 ±0.2	6.94 ±0.1
		CER_{val}	104.00 ±37.8	143.77 ±25.1	6.66 ±0.2	6.80 ±0.2
		STKW	74.67 ±4.8	140.50 ±16.9	7.77 ±0.4	7.73 ±0.3

Table 9. The comparison between the computing time of the early-stopping based on CER_{val} and the STKW early-stopping method of the architectures (Arch.) A_{15}, A_{14}, and A_{16} in Table 2 on RIMES. The values are in seconds per epoch. The loading (I/O) time of the validation set is 40 seconds. This time is not regarded in computing of CER_{val}. We used NVIDIA V100.

Arch.	CER_{val}	STKW
A_{14}	37.8	0.4
A_{15}	38.4	0.6
A_{16}	34.0	0.5

The computing time of the early-stopping method based on CER_{val} and the $SSKW$ early-stopping method are shown in Table 9 for the architectures (Arch.) A_{15}, A_{14}, and A_{16} in Table 2. The numerical values are in seconds per epoch. The loading (I/O) time of the validation set (40 seconds) is not considered in computing of CER_{val}. In comparison with the early-stopping method based on CER_{val}, the label-free STKW heuristic is at 60 to 90 times faster per epoch.

5 DISCUSSION

In this study, we explored three label-free early-stopping heuristics for convolutional neural networks. The results were compared to two well-known standard methods in the deep learning domain, i.e., based on the character-error rate on the validation-set CER_{val}, or, alternatively, on the CTC training loss. Both traditional methods require labels, for each of the instances in the validation data set.

Our exploration of alternative heuristics for early stopping started with SKW, the standard deviation of kernel-weight values. The first heuristic is the Cumulative Distribution of SKW (CD-SKW). This heuristic has the advantage of capturing the diversity of the distribution of weight values, which appears to increase over the epochs as a consequence of the increasingly refined weight updates during gradient descent. The results of CD-SKW were very promising, indicating that convenient stopping points could be found near the points that would have been obtained by the early-stopping method based on CER_{val} for the tested CNN models on the highly multilingual historical MkS data set, Fig. 1b. However, there appeared to exist a clear influence of the training schedule used, in this case, ADAM [15] where the learning rate changes in intervals of training epochs.

We have observed that a step-wise decrease in the learning rate determined by the applied learning-rate schedule causes a clear drop in the growth rate of the SKW value. This effect is not surprising because the evolution of the state of the neural network, including the kernel weights, slows down when the learning rate has a lower value. As a result, also CER_{val} as well as $Loss(train)$ show a reduction in changes around this point.

It is worth mentioning that the reported result for heuristic 1, CD-SKW, is on a realistic historical data set with various word-segmentation problems compared to the clean well-known standard benchmarks. Moreover, a significant boost in the accuracy of > 20% points is possible by applying a dictionary-based decoder such as a dual-state word-beam search (DSWBS [26],[1]). In this study we avoided the use of a dictionary in order to keep a strict focus on the shape classification and the associated CTC loss.

When there is no dynamic training schedule used, there is no clear 'knee' or bending point for SKW which we explore in heuristic 2. The results on two benchmark data sets indicated that the SKW values increase but the variation decreases during training. We measure the variation with the running standard deviation of SKW (SSKW). The evolution of SSKW showed similarity with CER_{val}. In heuristic 2, the training stops when SSKW values do not decrease in an interval of epochs. The results showed that the slight change in the size of this interval (15 vs 17 epochs) does not have a clear effect on CER_{test}. The larger window size of the running standard deviation (15 vs 10) gives a slightly better CER_{test} (%). The training process stops later when using heuristic 2, as compared to the use of CER_{val}.

The third heuristic is the moving standard deviation of total sum of SKW over epochs (STKW). This measure represents the evolution of the variability in the total sum of weight values over the epochs. Assuming that the

Table 10. The symbols used in the Algorithms

Type	Entry	discretion
Set	$data$	The training data set
	$layer$	All of the layers of the model
	KW	One-dimensional array with the kernel weight values of the model at the end of an epoch
Model	$model$	model
	$model_t$	The trained model at the iteration t
	$kernel\_weights$	Kernel weights
Scalar	t	The iteration counter
	NL	The number of the layers of the model
	nl	The counter for the layers, $nl = 1, ..., NL$
	ω	The window size to investigate the occurrence of an early stopping point in heuristic 1
	ω_1	The run-in stage window size
	ω_2	The window size of the running stdev. of the TKW values
	ω_3	The window size to investigate the occurrence of an early stopping point in heuristic 3
	p	The probability calculated by the Cumulative distribution function (CDF)
	τ	A predefined threshold for probability p
	SKW_t	The value of the stdev. of KW for the model at the epoch t
	T_{stop}	The stopping epoch
	TKW	The value of the sum of KW
	$STKW$	The value of moving stdev. of TKW of the ω_2 training iterations
	$STKW_t$	The $STKW$ at the time t
	i	The counter of the repetitive iteration with no decrease in the value of $STKW$
	j	The index of $STKW$
	$STKW_{min}$	The minimum $STKW$
Function	get_layers	Extracting the layers of a model
	$flatten$	The function collapsed the input array into one-dimensional array
	$\frown$	The concatenate function
	sd	Standard deviation
	$Initialize$	The function for initialization of the model
	CDF_ω	The Cumulative distribution function for the last ω epochs (equation 2)

training process approaches stability towards the end, it can be assumed that the running standard deviation of the total sum of weight values will decrease and stabilize. This property makes the heuristic interesting for use, requiring only a minor adjustment to the early-stopping rule. Computing the STKW heuristic is 60 to 90 times faster than the early stopping based on CER_{val} for each epoch. It demands additional training epochs, and CER_{test} is slightly higher. However, total duration will always be considerably shorter than in case of the standard approach.

The early stopping based on CER_{val} demands evaluating the model at each epoch on a holdout validation set. Although the single inference step can be quick, the computational cost of the evaluation of a network based on the *CER* is highly dependent on the size of the validation set. In the case of a large data set and a large number of epochs, this type of evaluation is time-consuming and a non-negligible part of the training process. One might say that early-stopping, based on the training loss presents similar benefits as three label-free early stopping methods regarding SKW or TKW. However, this method demands much more training epochs compared to the early stopping based on CER_{val} and also the label-free heuristics in which the learning rate is a constant value. Not using the plain training loss for early stopping has the advantage that a possible overfit can be avoided because our label-free heuristics do not entail task-specific constraints in the way each target label would have its effect via the measured loss.

The three label-free early-stopping heuristics have two major advantages: (1) The evaluation of a CNN model is not computationally intensive over a validation set per epoch; (2) All labeled data including the less frequent classes can be used as a training set.

Local minima are a pervasive problem in neural-network training. Overfitting is concerned with a model sticking to such a local minimum, at the expense of the patterns that are not well covered by this particular solution. Several remedies have been proposed in recent years. In [14], it is proposed to search for robust *flat minima* in the loss landscape, which is highly interesting but complicated. The detection of the presence of overly specific model solutions is currently mostly done by means of k-fold cross-evaluation and evaluating several architecture variants. A high variance of the performance criterion will be indicative of model sampling from diverse local minima. We assume that the smooth curve of SKW, in comparison to the -$LOSS(train)$ and CER_{val}- curves, helps to prevent overtrained solutions or local minima in the loss landscape by realizing a stable, well-timed early-stopping event.

6 CONCLUSIONS

In this study, we do not aim to surpass the character error rates that are reported in the state-of-the-art literature. Instead, we propose to develop label-free early-stopping methods for convolutional neural networks (CNNs). The suggested methods monitor the evolution of the standard deviation or total sum of the kernel weight values (SKW and TKW) during the training.

We compare the performance of the label-free early-stopping heuristics with two prevalent methods based on training loss and character error rate on the validation and test sets using 'difficult' historical word images of the MKS data set and two common benchmark data sets. Avoiding label spillage is essential for handwriting recognition in historical document collection because there is always a lack of sufficient labeled data, especially in the case of newly discovered script styles. This problem is exacerbated by the strongly skewed Zipf [39]-like distributions for the occurrence of character classes in texts.

The contributions of this study are:

(a) Introducing novel properties (SKW and TKW) for CNNs and characterizing its modification during the training process.
(b) Applying an adequate method based on SKW and TKW behaviors for terminating the training process;

(c) Comparing the occurrence of early-stopping points for methods based on the loss values on the model training set, the character error recognition on the validation set, and the proposed three early-stopping methods;
(d) Studying the role of the learning rate on four stopping criteria: the suggested SKW and TKW, character error recognition on the validation set, and the loss value of the model on the training set and showing the analogous effect of learning rate on the three criteria;
(e) Comparing the computational speed of three label-free suggested heuristics with the usual early-stopping method based on character error recognition on the validation set.

For future work, we plan to investigate other properties of the kernel weights during training, for different neural networks, especially Long short-term memory (LSTM [14]).

Algorithm 1: Heuristic 1: The CD-SKW early-stopping criterion

```
Input: $data, model, \tau, \omega$;
Result: $T_{stop}$
compute_SKW($model$)
    $layer \leftarrow get\_layers(model)$;
    $NL \leftarrow length(layer)$;
    $KW \leftarrow []$;
    for $nl \leftarrow 1$ to $NL$ do
        if $layer[nl]$ is $kernel\_weights$ then
            $F \leftarrow flatten(layer[nl])$;
            $KW \leftarrow KW \frown F$;
        end
    end
    $SKW = sd(KW)$;
    return $SKW$
main($data, model, \omega, \tau$)
    $t \leftarrow 0$;
    $model_t \leftarrow initialize(model)$;
    $p \leftarrow 0$;
    while $p > \tau$ do
        $t \leftarrow t + 1$;
        $model_t \leftarrow train(model_{t-1}, data)$;
        $SKW_t \leftarrow compute\_SKW(model_t)$;
        if $t > \omega$ then
            $p \leftarrow CDF_{\omega}(SKW_{t-\omega,t})$;
        end
    end
    $T_{stop} \leftarrow t$;
    return $T_{stop}$ ;
$T_{stop} \leftarrow main(data, model, \omega, tau)$;
```

7 ACKNOWLEDGMENTS

This work is part of the research program Making Sense of Illustrated Handwritten Archives with Project Number 652-001-001 financed by the Netherlands Organisation for Scientific Research (NWO). We would like to thank Dr. Andreas Weber for organizing a group of volunteers to annotate the manuscript of this project and for his tight cooperation with the University of Groningen. We thank the Center for Information Technology of the University of Groningen for providing access to the Peregrine high-performance computing cluster.

REFERENCES

[1] Mahya Ameryan and Lambert Schomaker. 2020. A High-Performance Word Recognition System for the Biological Fieldnotes of the Natuurkundige Commissie. In *Proceedings of the International Conference Collect and Connect: Archives and Collections in a Digital Age,*

Algorithm 2: Heuristic 3: The STKW early-stopping criterion

```
Input: $data, model, \omega_1, \omega_2, \omega_3$;
Result: $T_{stop}$
compute_TKW($model$)
    $layer \leftarrow get\_layers(model)$;
    $NL \leftarrow length(layer)$;
    $KW \leftarrow []$;
    for $nl \leftarrow 1$ to $NL$ do
        if $layer[nl]$ is $kernel\_weights$ then
            $F \leftarrow flatten(layer[nl])$;
            $KW \leftarrow KW \frown F$;
        end
    end
    $TKW = sum(KW)$;
    return $TKW$
main($data, model, \omega_1, \omega_2, \omega_3$)
    $t \leftarrow 0$;
    $i \leftarrow 0$;
    $j \leftarrow 0$;
    $model_t \leftarrow initialize(model)$;
    $STKW_{min} \leftarrow \infty$;
    while $i < \omega_3$ do
        $t \leftarrow t + 1$;
        $model_t \leftarrow train(model_{t-1}, data)$;
        if $t > \omega_1$ then
            $TKW_t \leftarrow compute\_TKW(model_t)$;
            $j \leftarrow j + 1$;
            if $j \geq \omega_2$ then
                $STKW_t \leftarrow sd([TKW_{t-\omega_2}, ..., TKW_t])$;
                if $STKW_t < STKW_{min}$ then
                    $STKW_{min} \leftarrow STKW_t$;
                    $i \leftarrow 0$;
                else
                    $i \leftarrow i + 1$;
                end
            end
        end
    end
    $T_{stop} \leftarrow t$;
    return $T_{stop}$ ;
$T_{stop} \leftarrow main(data, model, \omega_1, \omega_2, \omega_3)$;
```

Leiden, the Netherlands, November 23-24, 2020 (CEUR Workshop Proceedings, Vol. 2810), Andreas Weber, Maarten Heerlien, Eulàlia Gassó Miracle, and Katherine Wolstencroft (Eds.). CEUR-WS.org, Leiden, Netherlands, 92–103. http://ceur-ws.org/Vol-2810/paper8.pdf

[2] Mahya Ameryan and Lambert Schomaker. 2020. Improving the robustness of LSTMs for word classification using stressed word endings in dual-state word-beam search. In *17th Int. Conf. Frontiers in Handwriting Recognition*. IEEE, Dortmund, Germany, 13–18.

[3] Mahya Ameryan and Lambert Schomaker. 2021. A limited-size ensemble of homogeneous CNN/LSTMs for high-performance word classification. *Neural Computing and Applications* 33 (2021), 8615–8634.

[4] David Duvenaud, Dougal Maclaurin, and Ryan Adams. 2016. Early Stopping as Nonparametric Variational Inference. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 51)*, Arthur Gretton and Christian C. Robert (Eds.). PMLR, Cadiz, Spain, 1070–1077. https://proceedings.mlr.press/v51/duvenaud16.html

[5] Basilis Gatos, Georgios Louloudis, Tim Causer, Kris Grint, Verónica Romero, Joan Andreu Sánchez, Alejandro H. Toselli, and Enrique Vidal. 2014. Ground-Truth Production in the Transcriptorium Project. In *2014 11th IAPR International Workshop on Document Analysis Systems*. 237–241. https://doi.org/10.1109/DAS.2014.23

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press, London, England. http://www.deeplearningbook.org.

[7] Alex Graves. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, Berlin. https://doi.org/10.1007/978-3-642-24797-2

[8] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *Proceedings of the 23rd International Conference on Machine Learning* (Pittsburgh, Pennsylvania, USA) *(ICML '06)*. ACM, New York, NY, USA, 369–376. https://doi.org/10.1145/1143844.1143891
[9] William H. Greene. 1997. *Econometric Analysis* (3rd ed.). Macmillan Publishing Company, New York, USA.
[10] Gintautas Grigas and Anita Juskeviciene. 2018. Letter Frequency Analysis of Languages Using Latin Alphabet. *International Linguistics Research* 1, 1 (2018), 18–31.
[11] Emmanuèle Grosicki, Matthieu Carré, Edouard Geoffrois, Emmanuel Augustin, and Francoise Preteux. 2006. La campagne d'évaluation RIMES pour la reconnaissance de courriers manuscrits. In *Actes Colloque International Francophone sur l'Ecrit et le Document (CIFED'06)*. Fribourg, Switzerland, 61–66.
[12] Isabelle Guyon, Amir Reza Saffari Azar Alamdari, Gideon Dror, and Joachim M. Buhmann. 2006. Performance prediction challenge. In *In Proceedings of the International Joint Conference on Neural Networks (IJCNN 2006*. IEEE, Vancouver, BC, Canada, 2958–2965.
[13] David Haussler. 1990. Probably Approximately Correct Learning. In *Proceedings of the Eighth National Conference on Artificial Intelligence*. AAAI Press, University of California, USA, 1101–1108.
[14] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computing* 9, 8 (1997), 1735–1780.
[15] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]
[16] Denis Kwiatkowski, Peter C.B. Phillips, Peter Schmidt, and Yongcheol Shin. 1992. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics* 54, 1 (1992), 159–178. https://doi.org/10.1016/0304-4076(92)90104-Y
[17] Yann LeCun and Yoshua Bengio. 1998. Convolutional Networks for Images, Speech, and Time Series. In *The Handbook of Brain Theory and Neural Networks*, Michael A. Arbib (Ed.). MIT Press, Cambridge, MA, USA, 255–258.
[18] Radu Manuca and Robert Savit. 1996. Stationarity and nonstationarity in time series analysis. *Physica D: Nonlinear Phenomena* 99, 2 (1996), 134–161. https://doi.org/10.1016/S0167-2789(96)00139-X
[19] Nelson Morgan and Herve Bourlard. 1990. *Generalization and Parameter Estimation in Feedforward Nets: Some Experiments*. Vol. 2. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 630–637.
[20] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning* (Haifa, Israel) *(ICML'10)*. Omnipress, USA, 807–814.
[21] Maryam M Najafabadi, Flavio Villanustre, Taghi M Khoshgoftaar, Naeem Seliya, Randall Wald, and Edin Muharemagic. 2015. Deep learning applications and challenges in big data analytics. *Journal of Big Data* 2 (2015), 1–21.
[22] Lutz Prechelt. 2012. *Early Stopping-But When?* Springer Berlin Heidelberg, Berlin, Heidelberg, 53–67. https://doi.org/10.1007/978-3-642-35289-8_5
[23] M. B. Priestley and T. Subba Rao. 1969. A Test for Non-Stationarity of Time-Series. *Journal of the Royal Statistical Society: Series B (Methodological)* 31, 1 (1969), 140–149.
[24] Manel Rhif, Ali Ben Abbes, Imed Riadh Farah, Beatriz Martínez, and Yanfang Sang. 2019. Wavelet Transform Application for/in Non-Stationary Time-Series Analysis: A Review. *Applied Sciences* 9, 7 (2019), 1–22. https://doi.org/10.3390/app9071345
[25] Bastian Rieck, Matteo Togninalli, Christian Bock, Michael Moor, Max Horn, Thomas Gumbsch, and Karsten Borgwardt. 2019. Neural Persistence: A Complexity Measure for Deep Neural Networks Using Algebraic Topology. In *International Conference on Learning Representations*. ICLR2019, New Orleans, Louisiana, United State, 1–25. https://openreview.net/forum?id=ByxkijC5FQ
[26] Harald Scheidl, Stefan Fiel, and Robert Sablatnig. 2018. Word Beam Search: A Connectionist Temporal Classification Decoding Algorithm. In *The International Conference on Frontiers of Handwriting Recognition (ICFHR)*. IEEE Computer Society, Niagara Falls, New York, USA, 253–258.
[27] Joan Andreu Sánchez, Verónica Romero, Alejandro H. Toselli, and Enrique Vidal. 2014. ICFHR2014 Competition on Handwritten Text Recognition on Transcriptorium Datasets (HTRtS). In *2014 14th International Conference on Frontiers in Handwriting Recognition*. 785–790. https://doi.org/10.1109/ICFHR.2014.137
[28] Otto Neall Strand. 1974. Theory and methods related to the singular value expansion and Landweber's iteration for integral equations of the first kind. *SIAM J. Numer. Anal.* 11 (1974), 798–825.
[29] Seba Susan, Rohit Ranjan, Udyant Taluja, Udyant Shivang Rai, and Pranav Agarwal. 2019. Neural Net Optimization by Weight-Entropy Monitoring. In *Computational Intelligence: Theories, Applications and Future Directions - Volume II*, Nishchal K Verma and A. K. Ghosh (Eds.). Springer Singapore, Singapore, 201–213.
[30] Tom Viering and Marco Loog. 2021. The Shape of Learning Curves: a Review. *ArXiv* abs/2103.10948 (2021), 1–46.
[31] Walter Pitts Warren S. McCulloch. 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* 5 (1943), 99–115.
[32] Andreas Weber. 2019. Collecting Colonial Nature- European Naturalists and the Netherlands Indies in the Early Nineteenth Century. *Low Countries Historical Review* 134-3 (2019), 72–95.
[33] Andreas Weber. 2021. *The Routledge Handbook of Science and Empire*. Routledge, London and New York, Chapter Natural history collections and empire, 80–86.

[34] Andreas Weber, Mahya Ameryan, Katherine Wolstencroft, Lise Stork, Maarten Heerlien, and Lambert Schomaker. 2017. Towards a Digital Infrastructure for Illustrated Handwritten Archives. In *Final Conference of the Marie Skłodowska-Curie Initial Training Network for Digital Cultural Heritage, ITN-DCH 2017*, M. Ioannides (Ed.), Vol. 10605. Springer, Olimje, Slovenia, 155–166.
[35] P. Werbos. 1974. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* Ph. D. Dissertation. Harvard University.
[36] Wikipedia. 2022. *Letter frequency.* https://en.wikipedia.org/wiki/Letter_frequency
[37] Xue Ying. 2019. An Overview of Overfitting and its Solutions. *Journal of Physics: Conference Series* 1168 (feb 2019), 022022. https://doi.org/10.1088/1742-6596/1168/2/022022
[38] Changyong Yu, Xin Qi, Haitao Ma, Xin He, Cuirong Wang, and Yuhai Zhao. 2020. LLR: Learning learning rates by LSTM for training neural networks. *Neurocomputing* 394 (2020), 41–50. https://doi.org/10.1016/j.neucom.2020.01.106
[39] GK Zipf. 1935. *The psycho-biology of language : an introduction to dynamic philology.* Houghton, Mufflin, Oxford.

8 APPENDICES