Erschienen in: Symposium on Visualization in Data Science (VDS) at IEEE VIS 2018

Visualization-Assisted Development of Deep Learning Models in Offline Handwriting Recognition

Martin Schall*
University of Applied Sciences Konstanz
Siemens Postal, Parcel & Airport Logistics GmbH, Konstanz

Dominik Sacha†
Siemens Postal, Parcel & Airport Logistics GmbH, Konstanz

Manuel Stein‡
University of Konstanz

Matthias O. Franz§
University of Applied Sciences Konstanz

Daniel A. Keim¶
University of Konstanz

ABSTRACT

Deep learning is a field of machine learning that has been the focus of active research and successful applications in recent years. Offline handwriting recognition is one of the research fields and applications were deep neural networks have shown high accuracy. Deep learning models and their training pipeline show a large amount of hyper-parameters in their data selection, transformation, network topology and training process that are sometimes interdependent. This increases the overall difficulty and time necessary for building and training a model for a specific data set and task at hand. This work proposes a novel visualization-assisted workflow that guides the model developer through the hyper-parameter search in order to identify relevant parameters and modify them in a meaningful way. This decreases the overall time necessary for building and training a model. The contributions of this work are a workflow for hyper-parameter search in offline handwriting recognition and a heat map based visualization technique for deep neural networks in multi-line offline handwriting recognition. This work applies to offline handwriting recognition, but the general workflow can possibly be adapted to other tasks as well.

1 INTRODUCTION

Offline handwriting recognition [21, 27] is the transcription of natural text from images. It is an active field of research in deep learning [3, 23], machine learning and document analysis. *Connectionist Temporal Classification* [5] in combination with *Multi-Dimensional Long Short Term Memory* [6, 10] represent the current state-of-the-art in offline handwriting recognition for Latin and Arabic languages. Recent publications [1, 20] proposed to use a combined CNN-RNN model for offline handwriting recognition. *Multi-Dimensional Connectionist Classification* [22], an alignment and loss function for training deep neural networks for segmentation-free multi-line offline handwriting recognition was recently proposed.

Experiments and practical application have shown a high accuracy when applying deep learning, not only in offline handwriting recognition, especially when large data sets are available for training. Examples for the successful application of deep learning [3, 23] are found in e.g. machine translation [12, 30], computer vision [13, 29] and game AI [15, 24]. However, understanding why deep neural networks (DNNs) can outperform other traditional machine learning approaches as well as revealing how such complex models operate is an under-explored research area. To address these problems, researchers have successfully applied information visualization and visual analytics techniques to make DNNs transparent and "black box" processes interpretive. Visualization techniques have been proposed [25,31] in order to improve the understanding of DNNs in their domains. Developers of such complex DNNs leverage loss functions, quality metrics and visualization techniques to derive actionable insights to improve the model and their results. Adaptions concern different stages along a typical machine learning workflow, such as data-preparation, model-development, model-learning, or evaluation. Such a machine-learning workflow typically covers multiple iterations of tuning hyper-parameters, model structures, or evaluating outputs of the learning process. Optimizations are typically done with respect to a loss function that defines the optimization criteria for the problem, e.g. minimizing the mean squared error on a defined data set. Hyper-parameters and model structure are parameters of the model that typically cannot be automatically optimized regarding a loss function. Techniques for automatic hyper-parameter search are for example grid search and cross validation. This cannot be easily applied to offline handwriting recognition since training and testing of a single model is very time consuming. Furthermore, there are many hyper-parameters to tune while developing and building a DNN model and training pipeline. Those hyper-parameters affect the data loading, transformation, model topology and training process. In addition to being numerous do these hyper-parameters include dependencies to other hyper-parameters. Overall this increases the time necessary for hyper-parameter search and thus the needed time for building and training a DNN model that solves the task at hand.

In this paper, we address this challenge of finding hyper-parameters for data-preparation, model-development and model-learning for offline handwriting recognition by introducing a novel visual analytics-assisted workflow allowing inspection and inquiries in order to identify relevant hyper-parameters and adjust them in a meaningful way. We guide users through the hyper-parameter search enabling them to make informed decisions on the choice of hyper-parameters and further data preparation. Our contributed workflow is designed to proceed towards specific recommendations for actions by inquiries using statistical measures in combination with visual evaluations. Our proposed visualization is based on heat map visualizations and allows inspecting the DNNs output and alignment for individual samples and characters. This method allows to utilize expert human knowledge while defining and training deep neural networks for offline handwriting recognition. Figure 1 shows our visualization technique applied to a single character of a data sample as a teaser of the overall technique detailed in Section 6. Our pro-

*e-mail: martin.schall@htwg-konstanz.de
†e-mail: dominik.sacha@siemens.com
‡e-mail: stein@dbvis.inf.uni-konstanz.de
§e-mail: mfranz@htwg-konstanz.de
¶e-mail: keim@dbvis.inf.uni-konstanz.de

Figure 1: Example of our proposed visualization technique to a single character of a data sample. This visualization allows to quickly compare the expected and actual model prediction in relation to the neural network input. Please note that the heat map pixel size is different from the image pixel size because of spatial subsampling applied in the model.

posed workflow improves model results, reduces the time necessary for model training and makes data-preparation, model-development and model-learning more transparent and efficient.

2 Positioning of our Work

Recent work [2] has identified three tasks in *explainable AI*: understanding, debugging and refinement/steering. This work focuses on both debugging and refinement. *Debugging* focuses on identifying parts of the model that are defect and improve on those in order to achieve a satisfying model accuracy. *Refinement* or *steering* is the process of incorporating expert knowledge into the training process or improving the training process itself to facilitate faster training or higher accuracy.

A recent survey [11] provides an overview on how to integrate visual analytics with deep learning. Our approach can be categorized in this survey framework as follows. *Why?*: "Debugging & Improving Models" is the main task of this work, but the presented visualization can also be used for comparison of models given that the accuracy of them is similar enough. *What?*: "Individual Computational Units" in the output layers fits this work, as well as "Network Architecture" while testing the chosen hyper-parameters and data preparation. *When?* is most likely "After Training" to improve the next training if possible, but some faults can be improved by pausing and continuation of the training. *Who?* is the "Model Developers & Builders" in this work. *How?* are "Line Charts for Temporal Metrics" for general evaluation, e.g. viewing the error rate over training time. The visualization technique presented in this work is covered by "Instance-based Analysis & Exploration". *Where?* is the "Application Domains & Models" in this work.

3 Deep Learning Model

The task that our deep learning model [22] is designed to solve is called *Segmentation-Free Multi-Line Offline Handwriting Recognition* which is a "super-task" of *sequence labeling* [4] and generalizes it from 1-dimensional sequences to n-dimensional problems. Offline handwriting recognition is the transcription of handwritten text from an image to a machine-processable character sequence. An example of a well known data set for this task is the IAM Offline Handwriting DB [14]. Segmentation-free multi-line offline handwriting recognition is the task of transcribing multiple text lines at once without splitting of the input image into smaller parts such as lines, words or characters. Here, only the resulting transcribed character sequence explicitly consists of multiple lines. This allows to correctly transcribe paragraphs of text with overlapping lines and characters. Overlapping lines are frequently seen in handwritten texts, as seen in e.g. Figure 3. Our model is a deep neural network that was trained with the *Multi-Dimensional Connectionist Classification (MDCC)* [22] loss.

MDCC trains deep neural networks for segmentation-free multi-line offline handwriting recognition in a semi-supervised fashion. Only the correct character sequence is available as truth data. Spatial information such as the location, size or orientation of characters is not included in the truth data. Truth data only consists of ASCII, Unicode or otherwise encoded strings. Output of the deep neural network is a probability distribution with two spatial dimensions that gives the probabilities of each pixel of the input image being part of a certain character from the target alphabet. This probability distribution can later be decoded to obtain the most likely character sequence [22]. Since the type of the neural network output and truth data is different, a loss function cannot be directly applied. MDCC calculates the *alignment* of the truth character sequence over the pixel space, which is again the same type of probability distribution as the neural network but with the correct character sequence enforced. The alignment probabilities are, similar to the neural network prediction, continuous values and not discrete zero/one classifications. *Cross Entropy* is now applied as a loss function between the neural network output and the calculated alignment. Cross entropy minimizes the uncertainty with respect to the alignment probability distribution when the probability distribution estimated by the deep neural network is known. It reaches its minimum of zero in case both probability distributions are identical. Equation 1 defines the cross entropy loss for the probabilities estimated by the deep neural network and the alignment probabilities of any given sample from the data set. Variables p and c denote the pixel position and character respectively.

$$\mathrm{CE}(\mathrm{net}, \mathrm{align}) = -\sum_{p}\sum_{c} \mathrm{align}_p^c \times \ln(\mathrm{net}_p^c) \tag{1}$$

Calculating the alignment requires inference in a general graphical model (undirected and cyclic), which is computationally intractable [16, 18]. MDCC uses *Loopy Belief Propagation (LBP)* [16] on a *Conditional Random Field (CRF)* as an approximate inference algorithm to achieve practical runtimes. LBP on a CRF is an iterative approximate inference algorithm and thus introduces additional hyper-parameters to the system.

The overall model and pipeline for our multi-line offline handwriting recognition system is shown in Figure 2. The input into our deep learning model is a grayscale image of a block of multiple handwritten text lines. Several alternating layers of *Multi-Dimensional Long Short Term Memory (MDLSTM)* [8, 10], convolutions and spatial subsampling are applied in order to extract high-level features from this image. Transcription to characters is achieved by a final feedforward layer with Softmax activation function that can be decoded to a character sequence. The decoding mechanism is detailed in [22]. Table 1 outlines the network topology and hyper-parameters of our model.

Table 1: Topology of sequential layers and hyper-parameters of our deep neural network. $|C|$ is the number of characters within the target alphabet.

Layer type	Parameters
Image input	8-bit grayscale, 300dpi
Subsampling	Window 3x3, stride 3x3
MDLSTM	8 cells per direction, tanh activation
Subsampling	Window 2x2, stride 2x2
Feedforward	64 neurons, no bias, tanh activation
MDLSTM	44 cells per direction, tanh activation
Subsampling	Window 3x3, stride 3x3
Feedforward	172 neurons, no bias, tanh activation
MDLSTM	80 cells per direction, tanh activation
Feedforward	$\|C\|$ neurons, with bias, linear activation
Softmax	Pixel-wise Softmax activation function

Overall contains our deep learning model and MDCC several types of hyper-parameters, e.g. number of layers, number of neurons or convergence criteria for LBP. These hyper-parameters interact and need to be chosen reasonable in order to obtain a useful and optimized model for offline handwriting recognition.

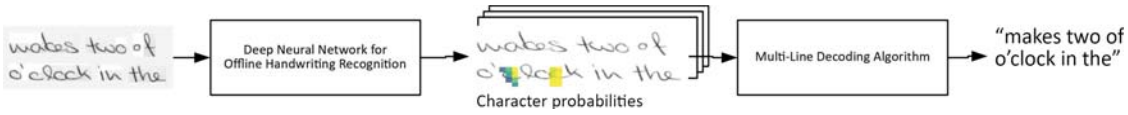

Figure 2: Overall machine learning model and pipeline for multi-line offline handwriting recognition using MDCC. The first rectangular box designates the deep neural network for which we optimize the hyper-parameters.

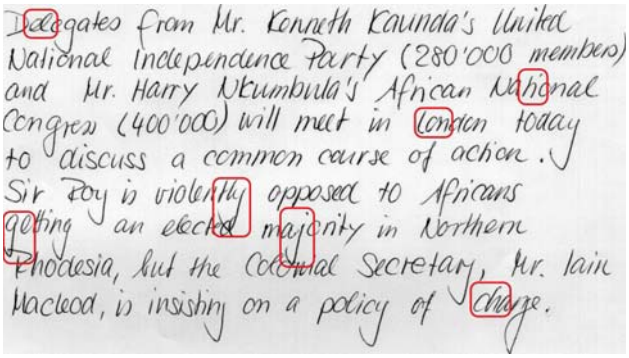

Figure 3: Example handwritten paragraph from the IAM Offline Handwriting DB [14] that shows overlapping lines and characters. Red markers were inserted by the authors to highlight problematic overlapping between characters or lines.

Metric for evaluation of a handwriting recognition system is the *Character Error Rate (CER)*. CER is the normalized Edit distance [28] between the transcribed character sequence and the truth character sequence. Lines are separated by one newline character for multi-line texts. Equation 2 defines the CER for the two strings. Note that CER has no upper bound and can be greater than 100.

$$\mathrm{CER}(\mathrm{transcr}, \mathrm{truth}) = 100 \times \frac{\mathrm{Edit}(\mathrm{transcr}, \mathrm{truth})}{|\,\mathrm{truth}\,|} \qquad (2)$$

4 Error Sources in Handwriting Recognition

Section 3 discussed the general task to be solved by our deep learning model, the model itself and the MDCC target function for training. The model performance for the given task and thus achieved accuracy is dependent on the hyper-parameters chosen by the user as well as other sources of error such as data preparation. It can be difficult to choose these hyper-parameters since they affect different properties of the model and sometimes influence each other. Automatic optimization of some of those hyper-parameters is possible (e.g. by cross-validation) in theory, but practically not feasible since one single training can run days to weeks even with GPU acceleration. It is thus in the interest of the model developer to make an informed decision on how to choose these parameters.

The following list gives potential reasons for unsatisfying accuracy while training a deep neural network with MDCC in order to solve the task of segmentation-free multi-line offline handwriting recognition on a given data set. The list identifies different types of potential reasons, namely problems with the data, hyper-parameters of the model, hyper-parameters of the training process and general errors.

1. **Data**
   (a) Too few data in general.
   (b) Too few data for outlier samples.
   (c) Truth data has systematic fault.
   (d) Truth data has individual samples wrong.
2. **Transformation**
   (a) Resolution of input image too low.
   (b) Resolution of input image too high.
3. **Network Topology**
   (a) Network topology not suitable for task.
   (b) Network capacity too small.
   (c) Subsampling within the network too large.
   (d) Subsampling within the network too small.
4. **Alignment**
   (a) LBP has not converged to stable configuration.
5. **Training Process**
   (a) Training is not finished yet.
   (b) Overfitting to training data.
   (c) Optimizer hyper-parameters are sub-optimal.
6. **General**
   (a) General configuration error.
   (b) Implementation bug.

Data: The first item identifies the data used for training and validation as a source of error. The data set used for training can be too small in general ("data is king") or can contain too few samples for specific outlier/infrequent cases. Note that handwriting recognition deals with inherently unbalanced data sets since not all characters of the alphabet occur with the same frequency in any given natural language. The data set can also be labeled with wrong "truth data" in a general systematic way (e.g. upper-/lower-case) or for individual samples (e.g. missing a word).

Transformation: Loading and transformation of the input image, namely the resolution of the loaded image is a source of error. A too low input resolution will obscure detail features of the handwritten text and thus make recognition harder. On the other hand does a too high resolution introduce further long-range dependencies within the image (e.g. the dot above a lower letter *i*) and thus make it more difficult to train a deep neural network that includes these long-range dependencies in its model.

Network Topology: The network topology of the deep neural network can also be a source of error. Its general configuration can be unsuitable for the task, e.g. a poor choice of non-linear activation functions. The capacity of the deep neural network can be too small which will prevent the network from using all features that are relevant to the task. The network capacity is controlled by the number of layers and number of neurons, both leading to more trainable parameters. The deep neural networks used in offline handwriting recognition use hierarchical subsampling that decreases the spatial resolution and thus allows learning of high-level feature representations. This subsampling can be chosen too large which will decrease the spatial resolution of the output further and a too low output resolution decreases the alignment possibilities and thus allows fewer variations (e.g. smaller or wider characters, text lines shifted left or right) in the handwriting. An extreme case is a output resolution that is smaller than the length of the text written in the image which means that it is not possible to calculate the correct alignment. There must be at least one pixel per character in the text to calculate a correct alignment and this alignment will not allow for

any variation in the handwriting. Choosing the subsampling too low will again introduce long-range dependencies that make the training process harder since features will be moved from the receptive field of a neuron to the outside of it. MDLSTM networks and its variants are capable of learning these long-range dependencies [9,10] but require more training iterations and data for it. Choosing the subsampling sizes and the resolution of the input image influence each other and should not be seen as independent parameters.

Alignment: LBP on a CRF is an iterative algorithm for approximate inference in general graphical models. Convergence to a stable configuration near the exact solution is not guaranteed, but often observed [17]. Running LBP for an appropriate number of iterations is important when training a deep neural network with MDCC since too few iterations will not lead to convergence to a stable configuration and too many iterations increase the runtime without providing benefits. This will then increase the overall runtime of the training process and during production deployment of the model.

Training Process: Further sources of error can be found in the general training process. It can simply be that the training has not finished yet and needs to run longer. Overfitting to the training set is also a rather common problem in machine learning. Overfitting is the adaptation of the model on the specific samples of the training set instead of adaption on the general concept of the data and the task at hand. It can be observed when the network capacity is too large with respect to the size of the training set. The optimizer used in training itself or its hyper-parameters can also be chosen sub-optimal, e.g. by a too high learning rate that leads to divergence of the error rate.

General: The last identified sources of errors in this case are general configuration errors, e.g. using the wrong target alphabet. Also possible are implementation bugs.

These identified reasons for an unsatisfying accuracy now need to be addressed by hyper-parameter search or other effective actions before or during training. Categories 1 (Data) through 3 (Network Topology) in general require stopping the current training run, modifications to mitigate the source of error and starting a new training run from scratch. Categories 4 (Alignment) and 5 (Training Process) can often be addressed by pausing the current training run and continuation after the according changes.

5 Workflow for Identifying Effective Actions

Section 4 identified the hyper-parameters and other potential sources of error during training of a deep neural network for offline handwriting recognition. The following section presents a novel workflow that the authors apply for a visual search for effective actions as outlined in Figure 4 in order to address the identified sources of error. The workflow is a combination of classical statistical evaluations and a visualization technique for offline handwriting recognition that will be detailed later in this paper, see Section 6.

Figure 4 illustrates a decision tree that guides the model developer through several inquiries in order to identify potential problems in data preparation or model hyper-parameters. The decision process starts in the top-left quadrant, which is designed to be performed only once per training. The remaining decision process in the bottom and right parts is repeated for several samples from the data set until the model developer reached a conclusion on how to proceed with the training. Figure 4 contains different types of nodes: plain rectangles for questions for inquiry, rectangles with rounded corners for visualizations and leaf nodes for suggestions for hyper-parameter modifications or other actions to mitigate errors. A trapezoid node is included to combine multiple options that are equally reasonable at this point in the workflow.

The workflow starts with Question A ("Validation CER satisfying?") to determine if the current *Character Error Rate (CER)* on the validation set is satisfying or not. The model can be used in its current state or trained further in hopes of even better accuracy, but modifications of the hyper-parameters are not necessary at this point. Testing the accuracy on different data sets often involves statistical tests with visual display, see Figure 5. Question B ("Validation CER still improving?") tests if the training process is still improving the model. In which case the training needs to run longer or maybe a change of the optimizer hyper-parameter are in order (e.g. slightly increase the learning rate). Question C ("Training CER lower?") is designed to identify overfitting of the model. If the accuracy on the validation set is not improving (or even worsening) but the accuracy on the training set is, it indicated over adaption to the training set. This can be countered by applying regularization methods such as Dropout [19,26]. Questions A through C can be answered by applying visualization techniques such as a line chart showing the change of error rates on the training and validation sets over time, as it is shown in Figure 5.

Question D ("All samples bad?") is were the workflow starts to get specific to offline handwriting recognition. This Question D and E ("Does it work with CTC?") are designed to test if the network topology itself is suitable to solve the task of offline handwriting recognition for single lines (sequence labeling). If all samples of the training set are recognized with a bad accuracy, we can now test if the network topology works when trained with *Connectionist Temporal Classification (CTC)* [5,7]. This requires prior segmentation of the images into individual lines or words, but is a robust way of testing the network topology. If the network performs equally bad with CTC, we can assume that either the network topology is unsuitable for the task or the data set is too small. Changing the network topology in this case involves most likely changing the layer types, number of layers or the type of activation function. Making the model larger in general (e.g. more neurons) is also a valid approach to this problem. Choosing the general network topology is often a question of the task at hand, e.g. CNNs for vision tasks and RNNs for machine translation tasks as well as the personal experience of the model developer. Increasing the model size is decided by observing the error convergence, see Figure 5, and increasing the model size if neither the training nor validation error reduces significantly over time.

Either way from Questions D or E is it now clear that the training process and deep neural network model themselves are suitable to solve the task of offline handwriting recognition. Questions A through E can also be answered once per training run since they are designed to test general properties of the model and data at hand. All further questions of the workflow require testing for individual samples and should be tested for multiple samples before reaching a conclusion.

Question F ("All characters bad?") requires the visualization of one bad sample from the training set. We propose a visualization for MDCC in Section 6 for this. Such a visualization must support answering the following questions:

- Were has the deep neural network predicted which characters?
- Are frequent or infrequent characters affected by problems?
- What is the relation between the input and output resolution of the model?
- Is the alignment encoding the correct character sequence?
- What is the difference between the prediction and alignment?

We apply this visualization technique to one bad sample in order to answer Question F. It is of interest if the difference between the prediction and alignment is only in frequent or infrequent characters or in both. Question F and I ("Which character affected?") answer this in combination. It is also necessary here to create a statistic of the characters within the alphabet beforehand. The statistic shows the frequency of occurrence of each character within the given corpus and can be visualized by a histogram plot. The frequency analysis of characters in the alphabet and Question I allow us to determine if frequent or infrequent characters are missing from the current sample. Missing frequent characters is likely a result of faults in

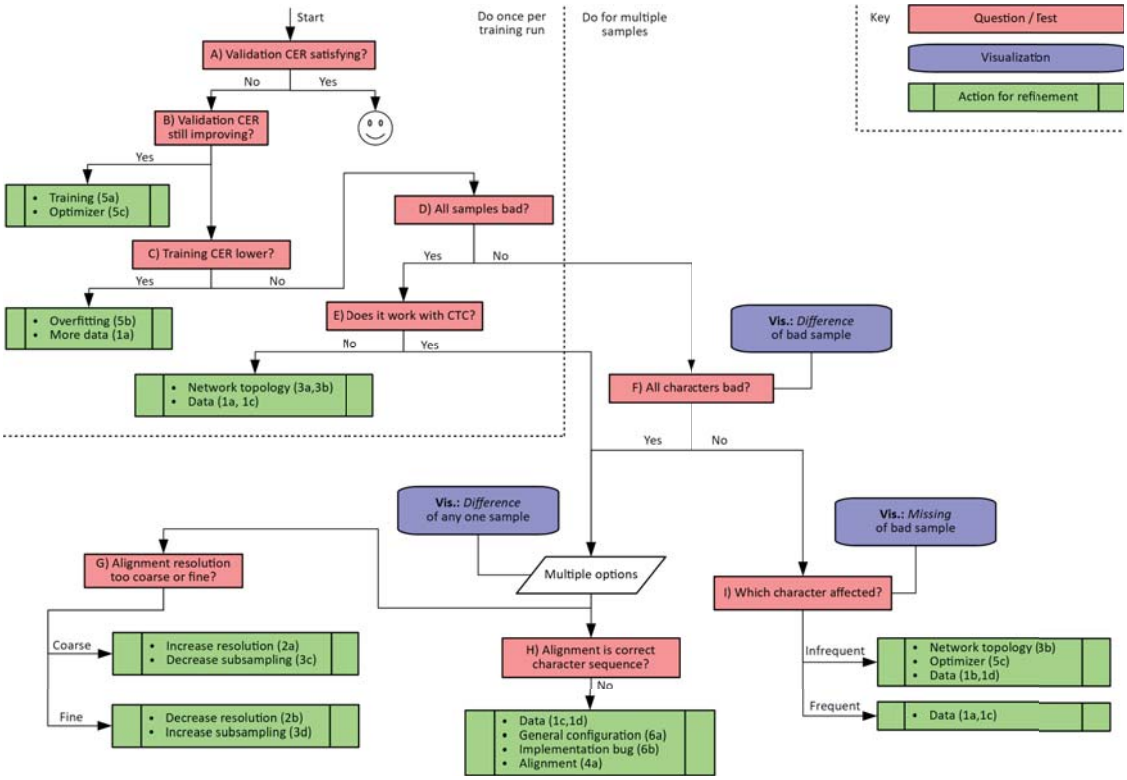

Figure 4: Workflow for identifying effective actions to mitigate sources of errors, e.g. hyper-parameters, in other to improve training in its current state. Nodes are of three different types: red for questions that need to be answered, blue for application of visualization techniques and green for suggested actions for refinement. Numbers in the leaf nodes refer to the source of error that these changes address.

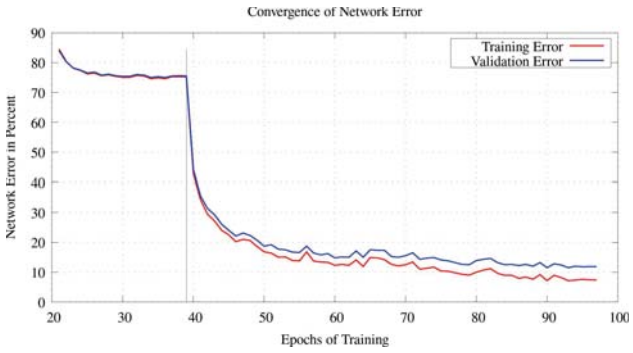

Figure 5: Line plot of the convergence of the model error over the training time. The x-axis is the training time in epochs, y-axis the model error in CER. The vertical gray line marks the switch from pre-training to training on the actual task at hand. See [22] for detailed information.

the truth data, but only in specific samples. Other questions in the workflow already determined that the network is capable of solving the task and only a few samples are showing a low accuracy, leading to the conclusion that missing frequent characters are caused by the truth data. Missing infrequent characters on the other hand can be caused by an unsuitable network topology such as incorrect choices of the non-linear activation functions. Other causes can be too few data samples containing the infrequent characters or a poor choice of the optimizer or its hyper-parameters, e.g. momentum leading to favoring frequent characters.

Question G ("Alignment resolution too coarse or fine?") and H ("Alignment is correct character sequence?") are based on the assumption that it has been shown that the network topology and data set are suitable for offline handwriting recognition, but all samples and/or all characters are equally affected by a low accuracy. Both questions are now asked and answered before coming to a conclusion on how to proceed with the training. Question G asks how the ratio between the resolution of the input image and the output prediction is. A too coarse ("blocky output") prediction can lead to less allowed variance and difficult to recognize details in the input image. It is then suggested to either increase the resolution of the input image or decrease the total subsampling applied by the deep neural network. A too fine output on the other hand makes it hard for the deep neural network to use long-range dependencies within the data. Decreasing the input resolution or increasing the subsampling is then in order.

Question H ("Alignment is correct character sequence?") is designed to determine if general problems with the data, model or implementation are occurring. This question tests if the calculated alignment represents the correct character sequence. This test can be done visually by "reading" the characters with the maximal probabilities in a left-right top-down order. If this is not the case, a problem with the truth data or configuration may be the cause. Convergence in LBP may be another cause which can be observed when the alignment probabilities show the character probabilities in "noisy clouds" rather than coherent blocks. Running LBP for more iterations per sample may be the solution in this case. The last possible reason in this case is an implementation error that prevents the alignment from working correctly in general.

6 VISUALIZATION FOR MDCC

Sections 4 and 5 discussed which hyper-parameters needs to be chosen for our deep neural network, potential sources of error and how to mitigate them by applying effective actions. Section 5 also discussed that a visualization is recommended for inspection of individual samples from the data set and the output that the deep neural network and alignment function produce.

We now present a novel application of visualization techniques for MDCC based on histograms and heat maps that allow us to quickly perform this inspection of individual samples from the data set. The first observation while designing the visualization technique is that the predicted probabilities from the deep neural network and the alignment probabilities from MDCC can be visualized per-character as a heat map over the pixel space. The grayscale input image is plotted in the background for orientation. The foreground heat map is then only partially plotted for all probabilities at most one standard deviation from the mean probability. Pixels of the prediction and alignment probabilities need to be duplicated for partially plotting over the input image since the resolution of the output is smaller than the input caused by subsampling in the model. These heat maps are the main part of our visualization and are shown in Figures 6 and 7.

Inspection of the heat maps allows the user to identify and distinguish several different sources of error. Errors in "truth data" can e.g. be identified by spotting discrepancies between the calculated alignment and the actual text written on the document. Errors in hyper-parameters related to the spatial subsampling and loading of images show in too fine or too coarse resolution of the deep neural network predictions and alignment. General sources of error or too low network capacities often result in random noise in the heat maps.

Note that our proposed visualization looks a bit similar to other methods [25, 31] of visualization of DNNs in the input pixel space. The referenced methods visualize the neurons receptive fields in the input space and thus make it possible to see to what type of input the neuron responds. Our visualization shows the network prediction partially plotted over the input image. This allows us to observe and inspect the overall model prediction, but not individual neurons.

The alphabet of a offline handwriting recognition system for Latin characters and English language consists of roughly 80 different characters and symbols, see size $|C|$ in Table 1. Showing one or more (as in Figures 6 and 7) such heat maps per character thus results in a high number of plots. The author's own usage showed that practical application and understanding of this depended on the experience of the user. We thus propose an ordering and filtering mechanism to quickly guide the user to the interesting heat maps within the set of all heat maps for all characters in the alphabet.

We propose to sort the heat maps per-character such that characters with interesting properties are shown first. There are several sort orders that can be applied here. Equations 3, 4 and 5 describe the metrics for these orders. Applying such an order means sorting the characters within one sample descending by the chosen metric. The top-n characters are then visually inspected by the user.

Difference as detailed in Equation 3 is based on the Cross Entropy loss function for training the neural network. It is thus very suitable for finding characters within the sample that that show a high discrepancy between the prediction from the deep neural network and and alignment function within MDCC. This makes it a suitable first entry point for inspecting the results that a model produces on a given data set.

$$\text{Difference}(\text{net}, \text{align}, c) = -\sum_p \text{align}_p^c \times \ln(\text{net}_p^c) \tag{3}$$

Ghosting, see Equation 4, orders characters first for which the prediction of the deep neural network contains positions of high probability but not the alignment function. This highlights parts of the sample were the model mistakenly detects not existing characters. *Ghosting* is similar to *false-positives* in normal classification tasks.

Figure 6: Heat maps of top-5 characters when using *Difference* as the sort criterion. *Difference* sort criterion is useful for quickly spotting the characters with the largest disparity between the network output and alignment. It is thus suitable for finding general improvements in hyper-parameters. Left column shows the predicted probabilities, right column the alignment probabilities.

$$\text{Ghosting}(\text{net}, \text{align}, c) = \sum_p \begin{cases} \text{net}_p^c & \text{if } \text{align}_p^c < \varepsilon \\ 0 & \text{else} \end{cases} \tag{4}$$

Missing, see Equation 5, is identical to *Ghosting* but with the neural network prediction and alignment function switched. It thus highlights positions were the alignment function expects this character, but the model did not detect it. *Missing* is thus *false-negatives* in classification.

$$\text{Missing}(\text{net}, \text{align}, c) = \sum_p \begin{cases} \text{align}_p^c & \text{if } \text{net}_p^c < \varepsilon \\ 0 & \text{else} \end{cases} \tag{5}$$

The combination of heat map visualizations and ordering of the characters within one sample allows now to guide the user to interesting characters within the inspected sample. Figure 6 show this visualization of one sample using the top-5 characters by the *Difference* sort order. Figure 7 shows the same sample but with top-5 characters according to the *Ghosting* sort order. These heat map plots are the main part of this proposed visualization technique.

Figure 6 prioritizes characters with a high difference between the network prediction and the alignment. This can be seen at character "c" and "a"which are only partly predicted or character "l"which is weakly (with low probability) predicted.

Figure 7 shows characters "d", "r"and "u"first, which are weakly predicted by the deep neural network but not at all occurring in the handwritten text. Characters "i"and "e"show one additional position each in which the character is not actually existing.

Ordering the characters within a sample and visualizing only top-n of them necessarily leads to many characters that will not be shown under this choice of subset. It is thus of interest for the user

Figure 7: Heat maps of top-5 characters when using *Ghosting* as the sort criterion. *Ghosting* sort criterion shows "false-positive" characters first, which occurs e.g. at truth data errors. Left column shows the predicted probabilities, right column the alignment probabilities.

if the top-n characters include all relevant characters or if the size n should be increased. We propose to visualize this in a second plot. This second plot is a histogram of all characters within the sample and their absolute value according to the chosen sort criterion. The histogram bars of the top-n characters are shown in another color than the not visualized characters. Figure 8 shows such a histogram plot. In this example we have used *Difference* as the sort order and visualized the top-5 characters. We can see that the top-5 characters include the relevant ones since there is a sharp drop from the 5th ("m") to the 6th ("o") character in this order.

We propose the following workflow: First let the user choose (with reasonable default values) the sort criterion and top-n size for the visualization. Afterwards, show the histogram as in Figure 8 to verify that these top-n characters are the interesting ones and change the size n if necessary. The last step is to inspect the heat map plots (Figure 6) in order to understand the output of the deep neural network and the alignment function of MDCC.

This visualization technique can be integrated with the workflow as described in Section 5 and Figure 4 to inspect individual samples from the data set.

7 Discussion

We have presented a novel visualization-assisted workflow for building models using Multi-Dimensional Connectionist Classification (MDCC). The workflow allows the user to perform a guided and visual hyper-parameter search while making informed decisions on possible sources of errors and actions as countermeasures. It is designed to be executed after the training process or during the training process and its goal is to improve the model and its hyper-parameters for the next training run.

Part of the workflow is the visualization and visual inspection of individual samples and the prediction that the model produces for these samples. We have proposed an application of heat maps partially superimposed on the input image as a visualization technique for MDCC. This allows to visually inspect the prediction of individual characters as well as testing general hyper-parameters of the model.

The workflow presented in this work is specific to MDCC, but the approach can possibly be generalized to other tasks as well. It is designed to identify typical machine learning problems during data preparation and training and provide suggestions for improvement. We leverage statistical techniques for identification of general machine learning problems and visual techniques for problems specific to MDCC. Suggestions for improvement are then provides in both case.

We can see the next steps for this line of work in three directions: generalization, integration with the training loop and integration with other visualization techniques. It is possible to generalize the proposed visualization-assisted workflow to other deep learning applications with multi-dimensional input since many error types, e.g. network topology, model size, data preparation or truth data, occur in both applications. It is then necessary to include other visualization techniques that are appropriate for the task at hand. A tight integration with the training loop further reduces the time necessary for inspection of the model. This increases the insight into the training process and increases the effectiveness of the training since hyper-parameters can be adapted quickly in this case. Integration with other visualization methods for DNNs, such as [25, 31], would allow to further refine the proposed workflow to provide improved suggestions for hyper-parameter refinements. As an example, visualizing the receptive field of the neurons of the last layer provides insight if character miss-classifications are caused by a too small model capacity or by wrong subsampling sizes. This allows the model developer to act accordingly on basis of the provided information.

Finally the authors would like to note that although this work is missing an user study for evaluation of our approach, it has been very helpful for us while working on models for offline handwriting recognition in the past. Our experience with this visualization-assisted workflow is that it reduced the time necessary for data preparation and transformation. It also further reduced the training time for models on new data since the workflow allows to quickly determine if a training run is "worth" continuing and if not, which parameters should be adjusted for the next run. This is crucial since a training run for a model that applies to offline handwriting recognition is in the time range of days to weeks even when utilizing fast multi-core processors and GPUs. We also at first tried visualizing all characters of a sample at once using the heat map based technique but without filtering. This was not very suitable for long term use since only experienced users could quickly discover interesting characters and thus the overall usage of the workflow and visualization took longer. This problem is mitigated by applying a sorting and filtering mechanism as detailed in Section 6.

Acknowledgments

The authors would like to thank the Siemens Postal, Parcel & Airport Logistics GmbH for funding this work. The authors would also like to thank Marc-Peter Schambach for his valuable feedback and suggestions while writing this paper.

References

[1] T. M. Breuel. High Performance Text Recognition using a Hybrid Convolutional-LSTM Implementation. 2017. doi: 10.1109/ICDAR.2017.12

[2] J. Choo. Visual Analytics for Explainable Deep Learning. 2018.

[3] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[4] A. Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, 2012.

[5] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *Proceedings of the 23rd international conference on Machine Learning*, pp. 369–376. ACM Press, 2006.

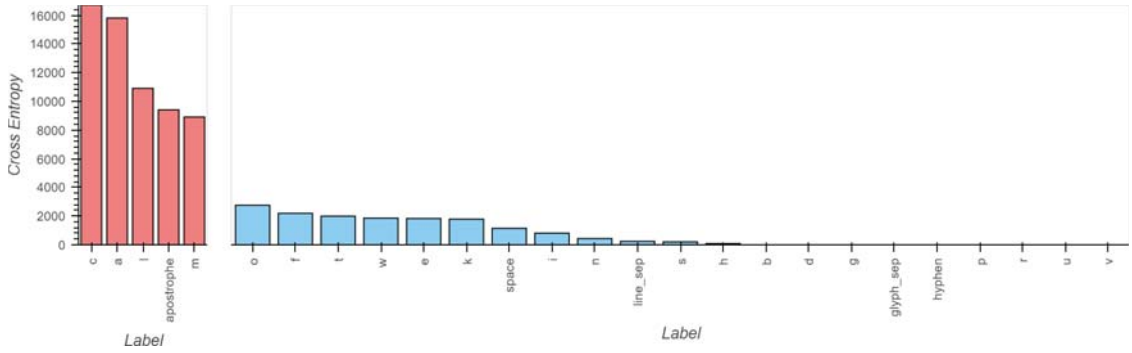

Figure 8: Histogram of the *Difference* for all characters within the given sample. It corresponds to the sample of Figure 6.

[6] A. Graves, S. Fernandez, and J. Schmidhuber. Multi-Dimensional Recurrent Neural Networks. Technical report, IDSIA/USI-SUPSI, 2007.

[7] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):855–868, 2009.

[8] A. Graves and J. Schmidhuber. Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks. In *Advances in Neural Information Processing Systems 21, NIPS'21*, pp. 545–552, 2008.

[9] K. Greff, R. K. Srivastava, J. Koutník, B. Steunebrink, and J. Schmidhuber. LSTM: A Search Space Odyssey. Technical report, IDSIA/USI-SUPSI, 2015.

[10] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[11] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau. Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers. pp. 1–20, 2018. doi: 10.1109/TVCG.2018.2843369

[12] M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, M. Hughes, and J. Dean. Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *arXiv*, pp. 1–16, 2016.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, pp. 1–9, 2012. doi: 10.1016/j.protcy.2014.09.007

[14] U. V. Marti and H. Bunke. The IAM-database: An English sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5(1):39–46, 2003.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with Deep Reinforcement Learning. pp. 1–9, 2013. doi: 10.1038/nature14236

[16] K. Murphy, Y. Weiss, and M. Jordan. Loopy-belief Propagation for Approximate Inference: An Empirical Study. *15*, pp. 467–475, 1999.

[17] K. P. Murphy. *Machine learning: a probabilistic perspective (adaptive computation and machine learning series)*. MIT Press, 2012.

[18] J. Pearl. Probabilistic Reasoning in Intelligent Systems, 1988. doi: 10.2307/2026705

[19] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour. Dropout Improves Recurrent Neural Networks for Handwriting Recognition. *Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR*, 2014-Decem:285–290, 2014. doi: 10.1109/ICFHR.2014.55

[20] J. Puigcerver. Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition? 2017. doi: 10.1109/ICDAR.2017.20

[21] K. M. Sayre. Machine recognition of handwritten words: A project report. *Pattern Recognition*, 5(3):213–228, 1973. doi: 10.1016/0031-3203(73)90044-7

[22] M. Schall, M.-P. Schambach, and M. O. Franz. Multi-Dimensional Connectionist Classification: Reading Text in One Step. In *Proceedings - 13th IAPR International Workshop on Document Analysis Systems, DAS 2018*, 2018.

[23] J. Schmidhuber. Deep Learning in Neural Networks: An Overview. Technical report, 2014. doi: 10.1016/j.neunet.2014.09.003

[24] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. doi: 10.1038/nature16961

[25] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *arXiv preprint arXiv:1312.6034*, pp. 1–8, 2013.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. doi: 10.1214/12-AOS1000

[27] T. Steinherz, E. Rivlin, and N. Intrator. Offline cursive script word recognition - a survey. *International Journal on Document Analysis and Recognition*, 2(2-3):90–110, 1999.

[28] R. A. Wagner and M. J. Fischer. The String-to-String Correction Problem. *Journal of the ACM*, 21(1):168–173, 1974.

[29] T. Weyand, I. Kostrikov, and J. Philbin. PlaNet - Photo Geolocation with Convolutional Neural Networks. Technical report, 2016.

[30] W. Zaremba and I. Sutskever. Learning to Execute. *ICLR*, pp. 1–25, 2015.

[31] M. D. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. *arXiv preprint arXiv:1311.2901*, 2013. doi: 10.1007/978-3-319-10590-1_53