Deep learning research over the past years has shown that by increasing the scope or difficulty of the learning problem over time, increasingly complex learning problems can be addressed. This principle has been described as Incremental learning (Elman, 1991). (Bengio et al., 2009) introduced the framework of Curriculum Learning. The central idea behind this approach is that a learning system is guided by presenting gradually more and/or more complex concepts. A formal definition is provided specifying that the distribution over examples converges monotonically towards the target training distribution, and that the entropy of the distributions visited over time, and hence the diversity of training examples, increases. Indiscriminate application of the curriculum learning principle however does not necessarily lead to improvement; it is essential therefore to identify which forms of curriculum learning yield substantial improvement. This work contributes to that aim by comparing three instantiations of curriculum learning, where it is found that only one of these yields notable improvements. An extension of incremental learning is also to let the learning task vary over time or, as a special case of transfer learning, to learn a suitable representation during learning (Pratt, 1993; Thrun, 1996; Moriarty, 1997; Gomez and Miikkulainen, 1997; de Jong and Oates, 2002).

**Incremental Sequence Learning** We study incremental learning in the context of sequence learning. The aim in sequence learning is to predict, given a step of the sequence, what the next step will be. An interesting challenge is that for most sequence learning problems of interest, the next step in a sequence does not follow unambiguously from the previous step. If this were the case, i.e. if the underlying process generating the sequences satisfies the Markov property, the learning problem would be reduced to learning a mapping from each step to the next. The dependency on the partial sequence received so far provides a special opportunity for incremental learning that is specific to sequence learning, as the steps in a sequence have a very particular relation; later steps in the sequence can only be learned well once the network has learned to develop the appropriate internal state summarizing the part of the sequence seen so far. This observation leads to the idea of learning to predict the earlier part of the sequences first. A prefix of a sequence is a consecutive subsequence (a substring) of the sequence starting from the first element. We define Incremental Sequence Learning as an approach to sequence learning where learning starts out by using only a short prefix of each sequence for training, and where the length of the prefixes used for training is gradually increased, up to the point where the complete sequences are used.

**Related Work** In (Bengio et al., 2009), a curriculum sequence learning example is provided where the subset of sequences used for training is gradually increased; this is analogous to one of the comparison methods used here. (Bengio et al., 2015) addresses the discrepancy between training and inference with training using scheduled sampling. (Zaremba and Sutskever, 2014) apply curriculum learning in a sequence-to-sequence learning context, where programs forming the training data are parameterized by the number of digits of the numbers used in the programs and the degree of nesting. While a number of different instantiations of incremental or curriculum learning have been described in the context of sequence learning, no clear guidance is available on which forms are effective. The particular form explored here of learning to predict the earlier parts of sequences first is straightforward, it makes use of the particular structure of sequence learning problems, and it is easy to implement; yet it has received very limited attention so far.

**MNIST Handwritten Digits as Pen Stroke Sequences** To obtain a sequence learning data set for evaluating Incremental Sequence Learning, we created a variant of the familiar MNIST handwritten digit data set (LeCun and Cortes, 2010) where each digit image is transformed into a sequence of pen strokes that could have generated the digit. See the long version of this abstract for details. The thinning operation discards pixels and therefore information; this implies that the sequence learning problem constructed here should be viewed as a new learning problem, i.e. performance on this new task cannot be directly compared to results on the original MNIST classification task.

**Network Architecture** We adopt the approach to generative neural networks described by (Graves, 2013) which makes use of mixture density networks, introduced by (Bishop, 1994). One sequence corresponds to one complete image of a digit, represented as a sequence of \( (dx, dy, cos, eod) \) tuples, and may represent one or more strokes; see previous section. The network has four input units, corresponding to these four input variables. To produce the input for the network, the \( (dx, dy) \) pairs are scaled to yield two real-valued input variables \( dx \) and \( dy \). The vari-
Incremental Sequence Learning

Figure 1. The original image (top left), thresholded image, thinned image, and actual extracted pen stroke image.

ables indicating the end-of-stroke (EOS) and end-of-digit (EOD) are binary inputs. Two hidden layers of LSTM units (Hochreiter and Schmidhuber, 1997) of 200 units each are used. The definition of the sequence prediction loss $L_P$ follows (Graves, 2013), with the difference that terms for the eod and for the L-∞ loss are included:

$$L(x) = \sum_{t=1}^{T} - \log \left( \sum_{j} \pi_{t,j} N(x_{t+1}|\mu_{t,j}, \sigma_{t,j}^2, \rho_{t,j}^2) \right)$$

$$- \begin{cases} \log eos_{t} & \text{if } (x_{t+1})_{3} = 1 \\ \log (1 - eos_{t}) & \text{otherwise} \end{cases}$$

$$- \begin{cases} \log eod_{t} & \text{if } (x_{t+1})_{4} = 1 \\ \log (1 - eod_{t}) & \text{otherwise} \end{cases} + \lambda ||w||_{\infty}$$

Incremental Sequence Learning and Comparison Methods

The baseline method is regular sequence learning: here, all training data is used from the outset. Incremental Sequence Learning: The prediction of later steps in the sequence can potentially depend on all preceding steps, and for some cases may only be learned once an effective internal representation has been developed that summarizes relevant information present in the preceding part of the sequence. The problem of learning to predict steps later on in the sequence is therefore potentially much harder than learning to predict the earlier steps. In Incremental Sequence Learning therefore, the length of sequences presented to the network is increased as learning progresses. Increasing training set size: following (Bengio et al., 2009), which describes curriculum sequence learning, we use subsets of the training data that grow in size. Increasing number of classes: The network is first presented with sequences from only one digit class; e.g. all zeros. The number of classes is increased until all 10 digits are represented in the training data. All three curriculum learning methods employ a threshold criterion based on the training RMSE.

Experimental results

Figures 2 shows a comparison of the results of the four methods. The baseline method (in red) does not use curriculum learning, and is presented with the entire training set from the start. Incremental Sequence Learning (in green) performs markedly better than all comparison methods. It reaches the best test performance of the baseline methods eight times faster; see the horizontal dotted black line. Moreover, Incremental Sequence Learning greatly improves generalization; on this subset of the data, the average test performance over 10 runs reaches 1.5 for Incremental Sequence Learning vs 4.7 for regular sequence learning, representing a reduction of the error of well over 60%. The variance of the test error is substantially lower than for each of the other methods, as seen in the performance graphs; and where the three comparison reach their best test error just before $4 \cdot 10^6$ and then begin to deteriorate, the test error for incremental sequence learning continues to steadily decrease over the course of the run.

To explain the dramatic improvement achieved by Incremental Sequence Learning, we consider two possible hypotheses:

$H1$: The number of sequences per batch is fixed (50), but the number of sequence steps or points varies, and is initially much smaller (2) for Incremental Sequence Learning. $H2$: Effectively learning later parts of the sequence requires an adequate representation of the preceding part of the sequence. To test $H1$, we design a second experiment where the batch size is no longer defined in terms of the number of sequences, but in terms of the number of points or sequence steps, where the number of points is chosen such that the expected total number of points for the baseline method remains the same. Figure 3 shows the results. This change removes the speedup during the earlier part of the run, and thus partially explains the improvements observed with Incremental Sequence Learning. However, the other observed improvements remain: Incremental Sequence Learning still features strongly improved generalization performance, has a much lower variance of the test error, and still continues improving at the point where the test performance of all other methods start deteriorating. In summary, the initially smaller and adaptive batch size of Incremental Sequence Learning explains part of the observed improvements, but not all. We therefore test to what extent hypothesis $H2$ plays a role. To see whether the ability to first learn a suitable representation based on the earlier parts of the sequences plays a role, we compare the situation where this effect is
rulled out. A straightforward way to achieve this is to use Feed-Forward Neural Networks (FFNNs). Figure 4 shows the results. As the figure shows, when using FFNNs, the advantage of Incremental Sequence Learning is entirely lost. This provides a clear demonstration that both of the hypotheses $H1$ and $H2$ play a role. Together the two hypotheses explain the total effect of the difference, suggesting that the proposed hypotheses are also the only explanatory factors that play a role.

**Results on the Full MNIST Pen Stroke Sequence Data Set** The results reported so far were based on a subset of 10000 training sequences and 5000 test sequences, in order to complete a sufficient number of runs for each of the experiments within a reasonable amount of time. Given the positive results observed with Incremental Sequence Learning, we now train this method on the full MNIST Pen Stroke Sequence Data Set (Experiment 1). Figure 3 shows the results. Compared to the performance of the above experiments, a strong improvement is obtained by training on this larger set of examples; whereas the best test error in the results above was slightly above 1.5, the test performance for this experiment drops below one; a test error of 0.972 on the full test data set is obtained.

**Transfer Learning** The first task considered here was to perform sequence learning: predicting step $t+1$ of a sequence given step $t$. To adequately perform this task, the network must learn to detect which digit it is being fed; the initial part of a sequence representing a 2 or 3 for example is very similar, but as evidence is growing that the current sequence represents a 3, that information is vital in predicting how the stroke will continue. Given that the network is expected to have built up some representation of what digit it is reading, an interesting test is to see whether it is able to switch to the task of sequence classification. The input presentation remains the same: at every time step, the recurrent neural network is fed one step of the sequence of pen movements representing the strokes of a digit. However, we now also read the output of the 10 binary class variable outputs. The target for these is a one-hot representation of the digit, i.e. the target value for the output corresponding to the digit is one, and all nine other target values are zero. Softmax is used on the output, and the sequence classification loss for the classification outputs is the cross entropy, weighted by a factor $\gamma = 10$. In the following experiments, the loss consists of the cross entropy classification loss to which optionally the earlier sequence prediction loss $L_p$ is added. Figure 5 shows the results; indeed the network is able to build further on its ability to predict pen stroke sequences, and learns the sequence classification task faster and more accurately than an identical network that learns the sequence classification task from scratch; in this first and straightforward transfer learning experiment based on the MNIST stroke sequence data set, eventually (not shown) a classification accuracy of 96.0% is reached.

**Conclusions** We have investigated Incremental Sequence Learning, and find that it strongly improves sequence learning performance, while two other instantiations of curriculum learning did not provide notable improvements. Incremental Sequence Learning reached the best test performance level of regular sequence learning eight times faster. It also reduced the test error of regular sequence learning by over
Incremental Sequence Learning

Experiment 3: FFNN, point-based batch size
Test error, average of 10 runs

Experiment 4: RNN on full MNIST Pen Stroke Sequence Data Set
Sequence-based batch size

Experiment 5: Transfer learning
from sequence prediction to sequence classification

Figure 4. Comparison of the test error of the four methods, averaged over ten runs.

Figure 5. Transfer learning: using the sequence prediction model as a starting point for sequence classification.

60%. Two other forms of curriculum sequence learning used for comparison did not display improvements compared to regular sequence learning.

We analyzed the cause of the observed speedup and performance improvements. By comparing RNNs with FFNN variants, evidence was found that the improvement in generalization performance is due to the specific ability of an RNN to build up internal representations of the sequences it receives, and that the ability to develop these representations is aided by training on the early parts of sequences first. The trained model was also used as a starting point for transfer learning from sequence prediction to sequence classification. The sequence prediction model was found to substantially speed up sequence classification learning.

We conclude that Incremental Sequence Learning provides a simple and easily applicable approach to sequence learning that was found to produce substantial improvements in both computation time and generalization performance. The dependency of later steps in a sequence on the preceding steps is characteristic of virtually all sequence learning problems. We therefore expect that this approach can yield improvements for sequence learning applications in general, and recommend its usage, given the exclusively positive results obtained with the approach so far.

Resources: The MNIST stroke sequence data set and code are available for download.

1. Resources

This extended abstract is a summary of a longer article, available as: https://arxiv.org/abs/1611.03068

Tensorflow implementation: https://github.com/edwin-de-jong/incremental-sequence-learning

The MNIST stroke sequence data set: https://github.com/edwin-de-jong/mnist-digits-stroke-sequence-data/wiki/MNIST-digits-stroke-sequence-data

Acknowledgements The author would like to thank Max Welling, Dick de Ridder and Michiel de Jong for valuable comments and suggestions on earlier versions.
References


