Automatic recognition of complementary strands: Lessons regarding machine learning abilities in RNA folding

Simon Chasles[1,2], François Major[1,2]

1 Institute for Research in Immunology and Cancer, Montréal, Québec, Canada
2 Department of Computer Science and Operations Research, Université de Montréal, Montréal, Québec, Canada

¤Current Address: 2900 boul Édouard-Montpetit, Montréal QC H3T 1J4, Canada
* francois.major@umontreal.ca

Abstract

Prediction of RNA secondary structure from single sequences still needs substantial improvements. The application of machine learning (ML) to this problem has become increasingly popular. However, ML algorithms are prone to overfitting, limiting the ability to learn more about the inherent mechanisms governing RNA folding. It is natural to use high-capacity models when solving such a difficult task, but poor generalization is expected when too few examples are available. Here, we report the relation between capacity and performance on a simpler related problem: determining whether two sequences are fully complementary. Our analysis focused on the impact of model architecture and capacity as well as dataset size and nature on classification accuracy. We observed that low-capacity models are better suited for learning with mislabelled training examples, while large capacities improve the ability to generalize to structurally dissimilar data. Given a more complex task like RNA folding, it comes as no surprise that the scarcity of usable examples hurdles the applicability of machine learning techniques to this field.

Author summary

We observe a growth in the use of neural networks to predict macromolecular structures by treating them as a collection of multiple binary classification tasks. When it comes to RNA structure prediction, we believe it is crucial to better understand the capabilities and behavior of neural networks. To this end, it has recently been proposed to use artificial data to study the performance of machine learning algorithms on specific tasks. In our study, we investigated the behavior of various neural network architectures on an RNA-related binary classification task using different artificial data configurations. We identified general trends of commonly used neural network architectures and analyzed the impact of different data distributions. We have found that the fixed-size input representation has limitations regarding length-wise extrapolation. We also obtained insights into the computing power required to handle particular data configurations in binary classification problems. We hope that our findings will promote greater caution and vigilance and lead to improved prediction accuracy when using neural networks for RNA folding tasks.

Introduction

Identifying potential structural candidates for a single RNA sequence is a computationally demanding task. The Zuker-style dynamic programming approach to fold an RNA sequence of length L without pseudoknots requires time complexity in $\mathcal{O}(L^3)$ [1] [2]. Algorithms that take into account pseudoknots are even more complex and have been reported to require significantly more computational power ranging from $\mathcal{O}(L^4)$ to $\mathcal{O}(L^6)$ [3] [4], or higher [5].

Machine learning (ML) algorithms offer an alternative to traditional methods for identifying RNA structural candidates. In particular, neural networks can compute structures in an end-to-end fashion, allowing for quick inference in a single feedforward pass, especially when running on `GPU` [6] [7]. However, the training phase can take several days, which can complicate software updates [8]. Regardless of the approach, predicting RNA structure requires significant computational resources due to the inherent complexity of RNA folding. As prediction methods must be at least as complex as the problems they aim to solve, it is important in the case of ML to avoid overfitting to this task.

To match the inherent complexity of RNA structure prediction, ML algorithms require high capacity, meaning they should be capable of learning a wide variety of mathematical functions [9]. Actually, because statistical learning relies heavily on training data, ML algorithms require high finite-sample expressivity to learn effectively [10]. However, high expressivity can lead to overfitting if the neural networks perform well on the training data without being able to generalize to structurally dissimilar testing data [11]. Therefore, it is crucial to balance expressivity with generalization to ensure accurate predictions on unseen data.

Several ML algorithms have been developed for RNA secondary structure prediction in recent years [6] [7] [12] [13] [14]. However, many of these algorithms are suspected of overfitting [15] [16] [17] and have limited ability to generalize across RNA families [18]. Generalization is crucial for accurate RNA structure prediction since known RNA structures only represent a small fraction of the entire RNA structure space. Prediction algorithms must be able to accurately predict structures for molecules that are similar to known structures (intrapolation) as well as those that are structurally dissimilar (extrapolation). Therefore, it is important to develop ML algorithms with stronger generalization properties to accurately predict RNA structures across a wider range of sequences and structures.

The aim of this study is to explore the performance and behavior of ML algorithms on a simple RNA-related task: determining if two RNA strands are fully complementary. Specifically, we examined the behavior of four families of neural networks with a focus on overfitting. We tackled three major challenges encountered when applying ML to RNA folding: (1) learning with mislabelled training examples (mislabels), (2) generalizing to structurally dissimilar data, and (3) training with limited examples [15] [19] [20] [21].

Our results indicate that low-capacity models are more effective for learning with mislabels, as they have the ability to ignore them. Conversely, high-capacity models demonstrate better generalization performance in some extrapolation contexts. On top of that, learning with few examples poses challenges for both low and high-capacity models, highlighting the importance of problem representation and architecture choice.

Materials and methods

Learning task

Given the RNA alphabet $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{U}\}$, complementary strands are those in which each nucleotide on one strand pairs with its Watson-Crick partner on the other strand (`A` with `U` and `C` with `G`). For example, the RNA strand 5'-`AGUCAG`-3' is complementary to 5'-`CUGACU`-3'. We defined the task of automatic recognition of complementary strands as a binary classification problem that involves comparing and determining whether pairs of RNA strands of the same length are complementary or not. The target is `True` if the strands are fully complementary and `False` otherwise.

To have a fixed-size noisy input and simulate the structure of a hairpin loop, we restricted the maximum length of an RNA strand to 8 nucleotides and inserted an apical loop of 4 random nucleotides between each pair of RNA strands. More specifically, given strands s, $\bar{s} \in \Sigma^L$ with $L \leq 8$ and $a \in \Sigma^4$, we represented the input as the sequence $0^{8-L} s a \bar{s} 0^{8-L}$, where 0 denotes zero-padding.

The target of the classification is $t = 1$ if for all $i \in \{1, ..., L\}$, s_i is paired with its Watson-Crick complement on the other strand, that is, if $s_i = c(\bar{s}_{L+1-i})$, where $c : \Sigma \to \Sigma$ is the Watson-Crick complementarity function defined as $c(\mathtt{A}) = \mathtt{U}$, $c(\mathtt{C}) = \mathtt{G}$, $c(\mathtt{G}) = \mathtt{C}$, and $c(\mathtt{U}) = \mathtt{A}$. Otherwise, the target is $t = 0$.

Each nucleotide was represented by a vector of size 4, and a word embedding layer was used to update these vectors during training. Therefore, the inputs are fixed-size real-valued matrices $x \in \mathbb{R}^{20 \times 4}$ and the outputs are real values $y \in (0, 1)$. The output can be interpreted as the probability of the RNA sequence being a positive example, which we also refer to as the positivity score.

The loss function used to train the models is the binary cross-entropy with adjustments made to account for varying ratios of positive and negative examples. Specifically, for a training dataset $\mathcal{D} = \left\{\left(x^{(i)}, t^{(i)}\right)\right\}_{i=1}^{N}$ with positive example ratio $\alpha = \frac{1}{N}\sum_{i=1}^{N} t^{(i)}$, the loss value for a model prediction $y^{(i)} = f\left(x^{(i)}\right)$ was computed by Eq (1).

$$l(y^{(i)}, t^{(i)}) = -\left[\left(\frac{1-\alpha}{\alpha}\right) t^{(i)} \log\left(y^{(i)}\right) + \left(1 - t^{(i)}\right) \log\left(1 - y^{(i)}\right)\right] \tag{1}$$

This correction ensured that the loss for positive examples was scaled up or down relative to the loss for negative examples, subject to the dataset's positive example ratio. For instance, if there were twice as many negative as positive examples in a dataset (i.e., $\alpha = 1/3$), the loss value for each positive example would be multiplied by a factor of 2, effectively balancing the contribution of positive and negative examples to the training loss. The parameter α could be controlled when creating synthetic datasets.

Artificial data

Our data generation process aimed to create diverse training and testing datasets that would enable us to evaluate our models in various scenarios. To achieve this, we introduced structural and statistical dissimilarities between the datasets, including differences in quality, sequence length and positivity rate. Mainly, we aimed at simulating the use of all known data (a training set) to infer predictions on a portion of the unseen data of interest (a corresponding testing set).

To produce a pair of training and testing datasets, we ensured that there was no overlap between the training and testing examples by randomly partitioning the set of all 4^L sequences of length L into two sets. For each example, we concatenated a

sequence $s \in \Sigma^L$ with a randomly generated apical loop a and a complementary sequence $\bar{s}$. Specifically, for positive examples, we set $\bar{s}$ to be the exact complement of s (denoted by s^*), while for negative examples, we chose $\bar{s}$ randomly in $\Sigma^L \setminus \{s^*\}$. We carefully controlled the ratio α of positive examples in the training sets and ensured that the testing sets had an equal number of positive and negative examples, thereby avoiding bias when measuring the performance on a test set.

To introduce structural dissimilarities, we varied the sequence length and the positive example ratio between the training and testing datasets. This allowed us to measure the extrapolation abilities of our models. Additionally, to control the quality of a training dataset, we introduced a proportion $\mu \in \left[0, \frac{1}{2}\right]$ of mislabelled examples, where a positive example could have label 0 with probability μ, and a negative example could have label 1 with probability μ. However, we ensured that all examples in the testing datasets were correctly labelled. By doing so, we tested the models' ability to ignore errors and avoid overfitting.

Overall, our data generation process produced realistic and diverse datasets, allowing us to evaluate our models under various conditions.

Performance measure

To evaluate the performance of our models, we measured their classification accuracy on the testing datasets using a classification threshold $\theta \in (0, 1)$ to distinguish between examples of class 0 and 1. Specifically, given a model $f(\cdot)$ and a dataset $\mathcal{D} = \{(x^{(i)}, t^{(i)})\}_{i=1}^N$, the accuracy of $f(\cdot)$ on $\mathcal{D}$ with threshold θ was calculated by Eq (2).

$$A\left(f(\cdot), \mathcal{D}, \theta\right) = \frac{1}{N} \sum_{i=1}^{N} \left(t^{(i)} \mathbb{1}\left\{ f\left(x^{(i)}\right) > \theta \right\} + \left(1 - t^{(i)}\right) \mathbb{1}\left\{ f\left(x^{(i)}\right) \leq \theta \right\} \right) \tag{2}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function that equals 1 if the condition inside the bracket is true, and 0 otherwise.

While threshold θ could be optimized through methods such as SVM [22] or maximizing accuracy on the training dataset [13], we set $\theta = 1/2$ to study the behavior of our neural networks independently of any additional optimization steps. We considered the mean accuracy over 50 simulations, each of which generated datasets as described above, trained a model as described in the next section, and evaluated its performance using the aforementioned accuracy metric.

Architectures and training

The four tested models are relatively small representatives of four types of neural networks that have been recently used to predict RNA structures [6] [13] [16]. All models have only three layers, where the first layer is unique to each architecture and the last two layers have the form `linear - batchnorm - ReLU - dropout - linear-sigmoid` [9] [23] [24] [25]. We used a dropout rate of 0.1 and a weight decay of 10^{-3} for all models, and the number of epochs was set to $\frac{8 \times 10^4}{N}$ to ensure that all models were trained for the same number of iterations, regardless of the number of training examples N. We used the `Adam` optimizer with a learning rate of 10^{-3} and default parameters in `PyTorch` [26] with a batch size of 256.

The four tested models and their capacity control are summarized in Fig 1. The multi-layer perceptron model (**MLP**) [9] has a first layer of the form `linear - batchnorm - ReLU - dropout`, and its capacity is controlled by the number H_1 of hidden units in the linear module. For the multi-head self-attention model (**Att**), the

first layer uses a skip connection of the form $h_1 = \texttt{conv}(x + \texttt{positional encoding})$ and $h_2 = \texttt{batchnorm}(h_1 + \texttt{dropout}(\texttt{multi-head self-attention}(h_1)))$, where the multi-head attention and positional encoding are described by Vaswani and co-workers [27]. The capacity of this layer is controlled by the number H_2 of heads in the multi-head attention, so the 1D-convolution uses a kernel of size 1 to project the 4-dimensional embedding into a H_2-dimensional input for the attention module. For the long-short term memory model (**LSTM**) [28] [29], the first layer is a one-layer bidirectional LSTM without dropout, and its capacity is controlled by the number H_3 of hidden units in the LSTM module. Finally, for the convolutional neural network (**CNN**) [11] [30], the first layer has the form `outer concatenation - 3x3conv - batchnorm - ReLU - dropout` with a stride and padding of size 1. The outer concatenation operation takes the 20×4 input matrix and returns a $20 \times 20 \times 8$ tensor, where the 8-dimensional vector at position i, j is the concatenation of the 4-dimensional encodings at positions i and j in the initial matrix. The capacity of the CNN is controlled by the number H_4 of feature maps produced by the convolution module.

Fig 1. The four neural network architectures. The 4 tested neural network architectures take nucleotide encodings as input and output the positivity score. All models have three layers, with the first layer being a characteristic layer, and the last two layers having the form Lin-`B`-`R`-`D`-Lin-σ, where Lin refers to a linear layer, Conv refers to a convolutional layer, Self-Att refers to a multi-head self-attention layer, BLSTM refers to a bidirectional long-short term memory layer, Pos Enc refers to positional encodings, and letters σ, `B`, `R` and `D` refer respectively to sigmoid activation, batch normalization, ReLU activation and dropout regularization. The capacity of the models is controlled by hyperparameters H_i and U_i.

To further manipulate the capacity of the models, we also varied the number U_i of hidden units in the second layer of our four architectures. Our main interest lies in the number C of parameters in each model, and more specifically in $\log_{10} C$. The actual number of heads, hidden units, and feature maps for each value of C is provided in Table 1. As discussed earlier, we trained 50 instances of each model on 50 different training datasets and reported the corresponding mean test accuracy. Our goal was to investigate how the test accuracy is affected by N and C for different values of L, μ and α.

Table 1. Hyper-parameters used to control the capacity of each model.

$\log_{10} C$	≈ 2.70	≈ 3.00	≈ 3.50	≈ 4.15	≈ 4.75	≈ 5.50
MLP (H_1, U_1)	(5, 5)	(10, 10)	(30, 30)	(80, 80)	(200, 200)	(500, 500)
Att (H_2, U_2)	(2, 9)	(4, 12)	(6, 24)	(12, 60)	(24, 120)	(80, 180)
LSTM (H_3, U_3)	(2, 5)	(3, 8)	(4, 18)	(10, 35)	(16, 78)	(48, 145)
CNN (H_4, U_4)	(1, 1)	(1, 2)	(2, 4)	(4, 9)	(8, 16)	(20, 36)

H_i refers to the capacity of the characteristic layer and U_i indicates the number of units in the second to last linear layer. The quantity $\log_{10} C$ gives an order of magnitude for the number C of trainable parameters.

Results and discussion

Learning with mislabels

The concept of mislabelling introduced in this section can be seen as a form of double standard. Our goal was to complicate the learning process deliberately by labelling some positive examples as class 0 and some negative examples as class 1. Thus, the term mislabelling is used because a ground truth classification is defined based on sequence complementarity. However, mislabelling on its own is not necessarily a problem for the learning process. A model $f(\cdot)$ that generalizes well on a completely mislabelled dataset could perform equally well on a correctly labelled dataset sampled from the same distribution by simply outputting $1 - f(\cdot)$. Therefore, the learning process is complicated when a proportion $\mu \in (0, 1)$ of the training dataset is mislabelled, creating a double standard as different pairs of complementary strands are labelled differently.

Moreover, since the labels can be interchanged without loss of generality, we focused on mislabelling probabilities $\mu \in \left(0, \frac{1}{2}\right)$. The special case where $\mu = \frac{1}{2}$ behaves as if the labels were randomly assigned [10]. In this study, we used a mislabelling probability of $\mu = 0.2$, which we find to be a good representative of the key findings presented in this section.

In comparison to the actual RNA structure prediction task, learning with mislabels can be seen as analogous to learning on a dataset that contains structures with non-canonical base pairs as well as structures within which non-canonical base pairs are ignored, even though such pairs exist and can be identified. Similarly, the same concept applies when training on a dataset aimed at predicting pseudoknots and triplets when these kinds of interactions are only reported for some structures but not for others. Without defining the structural elements that need to be predicted, situations of double standard can arise when certain types of base pairs can be labelled as present (positive) as well as absent (negative) in a single dataset.

We were interested in investigating how the number of parameters C and the number of training examples N affect the test accuracy of the models for the automatic recognition of complementary strands. In particular, we focused on fixed sequence length $L = 8$ and mislabelling probability $\mu = 0.2$. The results for the MLP and Att models are presented in Fig 2, where heatmaps depict the performance of the models for different values of C and N. Shades of red depict train accuracies; blue testing. See S1 Fig for the equivalent LSTM and CNN results. As expected, the accuracy increases with N, and overfitting behavior is observable for $\log_{10} C \geq 4.15$.

Fig 2. Performance of MLP and Att models when learning with mislabels. Train (red) and test (blue) mean accuracies over 50 simulations reported for MLP and Att models. Sequence length and mislabelling probability are respectively fixed to $L = 8$ and $\mu = 0.2$.

Remarkably, these results demonstrate the models' ability to handle mislabels, as they achieve a test accuracy of over 80% even when 20% of the training set is mislabelled. Moreover, the models can achieve near-perfect test accuracies as long as they are trained on a sufficiently large dataset. This finding suggests that models with relatively low capacity can still learn effectively when trained on low-quality high-quantity datasets.

Indeed, it appears that low-capacity models ($\log_{10} C \approx 3.5$) are more likely to achieve test accuracies above $1 - \mu$ than high-capacity models ($\log_{10} C \approx 5.5$). This is likely due to the fact that mislabels introduce irregularities into the sample space that are difficult for low-capacity models to account for. Low-capacity models tend to compute smoother functions than high-capacity models. They are thus less capable of

capturing the complex patterns that arise from mislabelling, especially in large training datasets. They have however enough capacity to estimate the function of interest, yielding test accuracies near 100% and train accuracies around $1 - \mu$.

On the other hand, high-capacity models can account for the irregularities introduced by the mislabels for larger training datasets, delaying the cross-over point of train and test accuracies to larger N. They thus require a broader view of the sample space to attain high test accuracies, coupled with sufficient regularization to prevent overfitting. The graphs in Fig 3 illustrate these behaviors. As in Fig 2, shades of red and blue depict train and test accuracies respectively.

Fig 3. Cross-over behavior when learning with mislabels. Influence of the training dataset size over train (Trn) and test (Tst) accuracies for low-capacity models (left) and high-capacity models (right). Sequence length is fixed to $L = 8$ and mislabelling probability is fixed to $\mu = 0.2$, with low capacity meaning $\log_{10} C \approx 3.5$ and high capacity meaning $\log_{10} C \approx 5.5$. Dotted lines indicate the 100% and 80% accuracy marks since 20% of the training examples are mislabelled.

However, it is important to note that these reported accuracies may be too optimistic because the models were trained and tested on sequences of the same length with the same ratios of positive and negative examples. Although no training sequence was repeated in the testing set, this setup only evaluates a model's ability to intrapolate within structurally similar data, but not to extrapolate to structurally dissimilar data. While this section highlights the effectiveness of low-capacity models in learning with mislabels, the following section presents a contrast as high-capacity models appear to be more suitable for generalizing to structurally dissimilar data.

Generalizing to structurally dissimilar data

In the context of the automatic recognition of complementary strands, the ability to generalize to structurally dissimilar data refers to the ability of a model to make accurate predictions over a subspace of the sample space that it has not or poorly seen during training. This means that models could be trained and tested on datasets containing different sequence lengths and positivity rates to evaluate their understanding of sequence complementarity. For example, a model can be trained on sequences of length 5 or 6 and then tested on sequences of length 8, or trained on datasets with few or a lot of positive examples and then tested on balanced datasets. This approach allows us to investigate how well a model can generalize and extrapolate its understanding of the problem.

In contrast, ML algorithms used for predicting actual RNA structure face different challenges. For instance, hardware limitations may restrict the maximum length of training sequences, but the model is still expected to predict the structure of longer sequences. The presence of non-canonical base pairs, pseudoknots, and base triples can increase the number of base pairs per nucleotide, making it difficult for models trained on sequences with a lower base pair density to generalize to more complex structures. Moreover, different RNA families may have varying frequencies of certain structural motifs [31], further complicating the generalization of models to unseen families. Despite these challenges, the ultimate goal of predicting RNA structure for all families remains the same, regardless of their structural similarity to previously known RNA structures.

To better visualize the impact of training example count N and the number of model parameters C on test accuracies for the automatic recognition of complementary strands, we report experiments on CNN and Att models. Specifically, the models were trained on correctly labelled sequences of length 6 and then tested on the full set of 4^8 sequences of length 8. We present heatmaps (Fig 4) that illustrate the performance of

the models as the number of parameters and training examples are varied. See S2 Fig for the equivalent MLP and LSTM results. We observed that higher model capacity tends to yield better performance, regardless of the number of training examples.

Fig 4. Performance of CNN and Att models in length-wise extrapolation context. Train (red) and test (blue) mean accuracies over 50 simulations reported for CNN and Att models. Models were trained on sequences of length 6 before being tested on sequences of length 8, without mislabelling in the training set.

Nevertheless, as we increase the capacity, signs of overfitting can still be observed. Notably, in the graphs presented in Fig 5, we observe signs of overfitting from the MLP model when $\log_{10} C \geq 4$, for both $(L, N) = (5, 500)$ on the left and $(L, N) = (6, 2000)$ on the right.

Fig 5. Influence of capacity in length-wise extrapolation context. Impact of the number of trainable parameters over train (Trn) and test (Tst) accuracies in length-wise extrapolation context. Results when training with 500 examples of length 5 is shown on the left and 2000 examples of length 6 on the right. Sequence length is set to 8 in the testing datasets and mislabelling probability is set to 0 in the training datasets. Dotted lines indicate the 100% and 90% accuracy marks to highlight acceptable test performance.

Interestingly, we observe that the MLP and CNN models appear to be better suited for extrapolation than the Att and LSTM models. The former can achieve test accuracies of over 90% with ease, while the latter struggle to do so. Additionally, the Att and LSTM models exhibit greater variability in their performances. For instance, the Att model of capacity $\log_{10} C \approx 4.15$ with $(L, N) = (6, 2000)$ has a mean test accuracy of 84.8% over 50 simulations. However, its best accuracy is over 99% and its worst accuracy is around 50%. In Fig 5, the error bars have been omitted to better highlight the performance of the four models.

Despite the impressive statistical performance of the MLP model, it fails to capture the nuances that make two RNA sequences complementary. Specifically, when trained on 500 sequences of length 5, the MLP model tends to mistake a negative example with mismatches in the upper three base pairs for a positive example. The MLP model with $\log_{10} C \approx 3.5$ achieves a statistical accuracy of 98.1%, yet it incorrectly classifies the sequence `ACGUACGUGAAAACGUAGCA` as a positive example with a mean positivity score of 0.9810 (Fig 6). Interestingly, all other models exhibit a similar trend, as they assign a comparable (albeit slightly lower) mean positivity score to this negative sequence as they do to its corresponding positive sequence.

Fig 6. Limits to length-wise extrapolation with zero-padding and fixed-size inputs. Distribution of model outputs (positivity score) on specific sequences of length 8 when trained on 500 sequences of length 5. The first sequence is a positive example and the second and third ones are the same positive example where 3 mismatches have been introduced respectively at base pair positions 6 to 8 and 2 to 4. We use the best tested capacity for each model so $\log_{10} C \approx 3.5$ for the MLP and $\log_{10} C \approx 5.5$ for all other models. The classification threshold is represented by the red dotted line.

Furthermore, the negative example with mismatches in the lower five base pair positions is correctly classified most of the time by all models. This suggests that the use of zero-padding to have fixed-size intputs limits the length-wise extrapolation abilities of ML models to the nucleotide positions seen during training. The MLP model seems to be particularly affected by the zero-padding, while the LSTM tends to produce

lower positivity scores for the negative sequence with mismatches in the upper three base pair positions, probably due to its sequential calculation.

It is worth noting that all results presented so far were obtained by training and testing on datasets with an equal number of positive and negative examples. However, we can manipulate the proportion α of positive examples in the training dataset while still testing on a balanced dataset. Despite the loss function correction that accounts for an equal number of positive and negative examples, when the concentration of positive examples is too low or too high, all four models perform poorly (Fig 7); even when classification threshold θ is equal to α.

Fig 7. Evolution of performance in the extrapolation context of various positivity rates. Influence of the concentration of positive examples in the training set on the train (Trn) and test (Tst) accuracies for low-capacity models (left) and high-capacity models (right). Parameters were fixed to $(N, L, \mu) = (500, 6, 0)$. Dotted lines indicate the 100% and 90% accuracy marks to highlight acceptable test performance.

If we aim to extrapolate accurately without mislabels, high-capacity models are favorable, which means that the training datasets must be sufficiently large to support this level of expressiveness. Moreover, when we want to intrapolate accurately but are willing to tolerate some mislabelling, the models require a substantial number of training examples before they can disregard the errors. In the next section, we will address the challenge of learning from a small training dataset while also consolidating the insights from the previous two sections on capacity.

Learning with few training examples

Here we focus on learning with small training datasets, but this at the same time provides us with an opportunity to explore how all our initial challenges interact with each other. Thus, it is an important section that consolidates the insights gained from the previous sections.

We can note from last sections that learning with high capacity is advantageous for extrapolating without mislabels, while learning with low capacity is necessary for intrapolating with mislabels. However, when both challenges are combined, model behaviors become more complex.

To illustrate this complexity, we present heatmaps showing the behavior of the four tested architectures when trained with various quantities of sequences of length 6 with a mislabelling probability of 20% and a positivity ratio of 40% (Fig 8), the later being an arbitrary value to further challenge the extrapolation abilities of the models. The test accuracies are reported over the whole set of sequences of length 8 with as many positive and negative examples. See S3 Fig for the corresponding train accuracies. These heatmaps demonstrate that the ability to ignore errors is conserved even when extrapolating, provided enough training examples are supplied. However, the required capacity to maximize generalization performance varies among the models and can be influenced by the number of training examples. For instance, the MLP and CNN require low capacity to attain their best generalization performance, while the Att and LSTM require high capacity. Additionally, the CNN's best generalization performance requires high capacity with fewer training examples, but low capacity with more training examples.

However, achieving acceptable performances with test accuracy over 80% requires a substantial number of training examples ($N \geq 2000$) when extrapolating with mislabels. When only a few training examples are available ($N \leq 500$), test accuracies over 80% become challenging to achieve. Fig 9A shows the influence of capacity on the accuracies

Fig 8. Performances for all models when learning with mislabels in extrapolation context. Test accuracies for all models when tested on sequences of length 8 after being trained on sequences of length 6 with 20% mislabelled training examples and 40% positivity rate.

when trained on 500 sequences of length 6 with 20% mislabelling and 40% positivity rate, tested on the whole balanced set of correctly labelled sequences of length 8. It appears that variations in capacity have a low impact on the model's performance, although a general slight improvement can be observed as capacity increases.

In order to gain more insight into the behavior of the four models when dealing with few or many training examples, we compared their distributions of performances (Fig 9B). To produce this figure, we used the capacities that maximize accuracy based on the heatmaps presented in Fig 8. As a result, a wide range of capacities was needed for the models, with the MLP using $C \approx 3500$ and the CNN using $C \approx 15\ 000$ when trained on many examples. On the other hand, the Att and LSTM required much higher capacities, with the Att using $C \approx 60\ 000$ and the LSTM using $C \approx 300\ 000$ regardless of the number of training examples. Analyzing these distributions, we observe that the MLP model appears to be the most suitable for extrapolating with mislabels since its test accuracies have a higher mean and lesser variance than the other three models.

Fig 9. Limits when learning with few examples with mislabels in extrapolation context. (A) Influence of capacity on train (Trn) and test (tst) accuracies. The training sets are unbalanced and mislabelled as parameters (N, L, μ, α) are fixed to $(500, 6, 0.2, 0.4)$. The testing sets on the opposite are balanced and correctly labelled with parameters (N, L, μ, α) being fixed to $(4^8, 8, 0.0, 0.5)$. (B) With the same parameters, distributions of test accuracies over 50 simulations are reported, with models being trained on 500 examples (e.g. MLP$^-$) or 4000 examples (e.g. MLP$^+$). The capacity used for each situation is the best capacity achieved on the test sets with respect to the heatmaps in Fig 8. Dotted lines indicate the 100% and 80% accuracy marks since 20% of the training examples are mislabelled.

Overall, this section highlights the importance of considering multiple challenges simultaneously when measuring the impact of capacity and training dataset size on the generalization performance of a variety of ML models. It also underscores the need for a better understanding of the trade-offs involved in choosing appropriate models and capacity for specific tasks, especially when dealing with small training datasets.

Conclusion

In conclusion, the use of statistical learning through neural networks holds great promise for gaining insight into the complex mechanisms that govern RNA folding. Even though over-parameterized models may be adequate for learning useful representations [32] [33], the fact remains that the quality of a ML model highly depends on the data on which it has been trained. Our study highlights three main challenges researchers face when working with current RNA structure data and provides suggestions for overcoming them.

Specifically, when dealing with mislabels, low-capacity models may be preferable, as long as enough training examples are provided. Moreover, for tasks that require extrapolating to structurally dissimilar data, high-capacity models may provide better performance. Finally, we recommend exploring the behavior of a variety of neural networks on synthetic data to better understand their specific risks and benefits in

predicting RNA structure. Overall, by addressing these challenges, machine learning could provide valuable insights into RNA folding and contribute to the development of new approaches for studying biological systems involving RNA.

Acknowledgments

We would like to express our sincere gratitude to Sébastien Lemieux and all the participants of the Computational Approaches to RNA Structure and Function conference held in Benasque, Spain in 2022 for their valuable suggestions and feedback.

References

1. Zuker M, Stiegler P. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. Nucleic acids research. 1981;9(1):133–148.

2. Hofacker IL, Fontana W, Stadler PF, Bonhoeffer LS, Tacker M, Schuster P, et al. Fast folding and comparison of RNA secondary structures. Monatshefte fur chemie. 1994;125:167–167.

3. Rivas E, Eddy SR. A dynamic programming algorithm for RNA structure prediction including pseudoknots. Journal of molecular biology. 1999;285(5):2053–2068.

4. Condon A, Davy B, Rastegari B, Zhao S, Tarrant F. Classifying RNA pseudoknotted structures. Theoretical Computer Science. 2004;320(1):35–50.

5. Marchand B, Will S, Berkemer S, Bulteau L, Ponty Y. Automated design of dynamic programming schemes for RNA folding with pseudoknots. In: WABI 2022 - 22nd Workshop on Algorithms in Bioinformatics. Potsdam, Germany; 2022.Available from: `https://hal.inria.fr/hal-03676377`.

6. Chen X, Li Y, Umarov R, Gao X, Song L. RNA secondary structure prediction by learning unrolled algorithms. arXiv preprint arXiv:200205810. 2020;.

7. Fu L, Cao Y, Wu J, Peng Q, Nie Q, Xie X. UFold: fast and accurate RNA secondary structure prediction with deep learning. Nucleic acids research. 2022;50(3):e14–e14.

8. Shen T, Hu Z, Peng Z, Chen J, Xiong P, Hong L, et al. E2Efold-3D: End-to-End Deep Learning Method for accurate de novo RNA 3D Structure Prediction. arXiv preprint arXiv:220701586. 2022;.

9. Goodfellow I, Bengio Y, Courville A. Deep learning. MIT press; 2016.

10. Zhang C, Bengio S, Hardt M, Recht B, Vinyals O. Understanding deep learning (still) requires rethinking generalization. Communications of the ACM. 2021;64(3):107–115.

11. LeCun Y, et al. Generalization and network design strategies. Connectionism in perspective. 1989;19(143-155):18.

12. Zakov S, Goldberg Y, Elhadad M, Ziv-Ukelson M. Rich parameterization improves RNA structure prediction. Journal of Computational Biology. 2011;18(11):1525–1542.

13. Singh J, Hanson J, Paliwal K, Zhou Y. RNA secondary structure prediction using an ensemble of two-dimensional deep neural networks and transfer learning. Nature communications. 2019;10(1):5407.

14. Wang L, Liu Y, Zhong X, Liu H, Lu C, Li C, et al. DMFold: A novel method to predict RNA secondary structure with pseudoknots based on deep learning and improved base pair maximization principle. Frontiers in genetics. 2019;10:143.

15. Rivas E, Lang R, Eddy SR. A range of complex probabilistic models for RNA secondary structure prediction that includes the nearest-neighbor model and more. RNA. 2012;18(2):193–212.

16. Sato K, Akiyama M, Sakakibara Y. RNA secondary structure prediction using deep learning with thermodynamic integration. Nature communications. 2021;12(1):941.

17. Zhao Q, Zhao Z, Fan X, Yuan Z, Mao Q, Yao Y. Review of machine learning methods for RNA secondary structure prediction. PLoS computational biology. 2021;17(8):e1009291.

18. Szikszai M, Wise M, Datta A, Ward M, Mathews DH. Deep learning models for RNA secondary structure prediction (probably) do not generalize across families. Bioinformatics. 2022;38(16):3892–3899.

19. Flamm C, Wielach J, Wolfinger MT, Badelt S, Lorenz R, Hofacker IL. Caveats to deep learning approaches to RNA secondary structure prediction. Biorxiv. 2021; p. 2021–12.

20. Burley SK, Bhikadiya C, Bi C, Bittrich S, Chen L, Crichlow GV, et al. RCSB Protein Data Bank: Celebrating 50 years of the PDB with new tools for understanding and visualizing biological macromolecules in 3D. Protein Science. 2022;31(1):187–208.

21. Danaee P, Rouches M, Wiley M, Deng D, Huang L, Hendrix D. bpRNA: large-scale automated annotation and analysis of RNA secondary structure. Nucleic acids research. 2018;46(11):5381–5394.

22. Smola AJ, Schölkopf B. A tutorial on support vector regression. Statistics and computing. 2004;14:199–222.

23. Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning. pmlr; 2015. p. 448–456.

24. Nair V, Hinton GE. Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10); 2010. p. 807–814.

25. Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:12070580. 2012;.

26. Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980. 2014;.

27. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. Advances in neural information processing systems. 2017;30.

28. Hochreiter S, Schmidhuber J. Long short-term memory. Neural computation. 1997;9(8):1735–1780.

29. Sak H, Senior AW, Beaufays F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: INTERSPEECH; 2014. p. 338–342.

30. Dumoulin V, Visin F. A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:160307285. 2016;.

31. Moore PB. Structural motifs in RNA. Annual review of biochemistry. 1999;68(1):287–300.

32. Belkin M, Hsu D, Ma S, Mandal S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. Proceedings of the National Academy of Sciences. 2019;116(32):15849–15854.

33. Peters E, Schuld M. Generalization despite overfitting in quantum machine learning models. arXiv preprint arXiv:220905523. 2022;.

Supporting information

S1 Fig. Performance of LSTM and CNN models when learning with mislabels. Train (red) and test (blue) mean accuracies over 50 simulations reported for LSTM and CNN models. Sequence length and mislabelling probability are respectively fixed to $L = 8$ and $\mu = 0.2$.

S2 Fig. Performance of MLP and LSTM models in length-wise extrapolation context. Train (red) and test (blue) mean accuracies over 50 simulations reported for MLP and LSTM models. Models were trained on sequences of length 6 before being tested on sequences of length 8, without mislabelling in the training set.

S3 Fig. Training performances for all models when learning with mislabels in extrapolation context. Train accuracies for all models when trained on sequences of length 6 with 20% mislabelled training examples and 40% positivity rate.

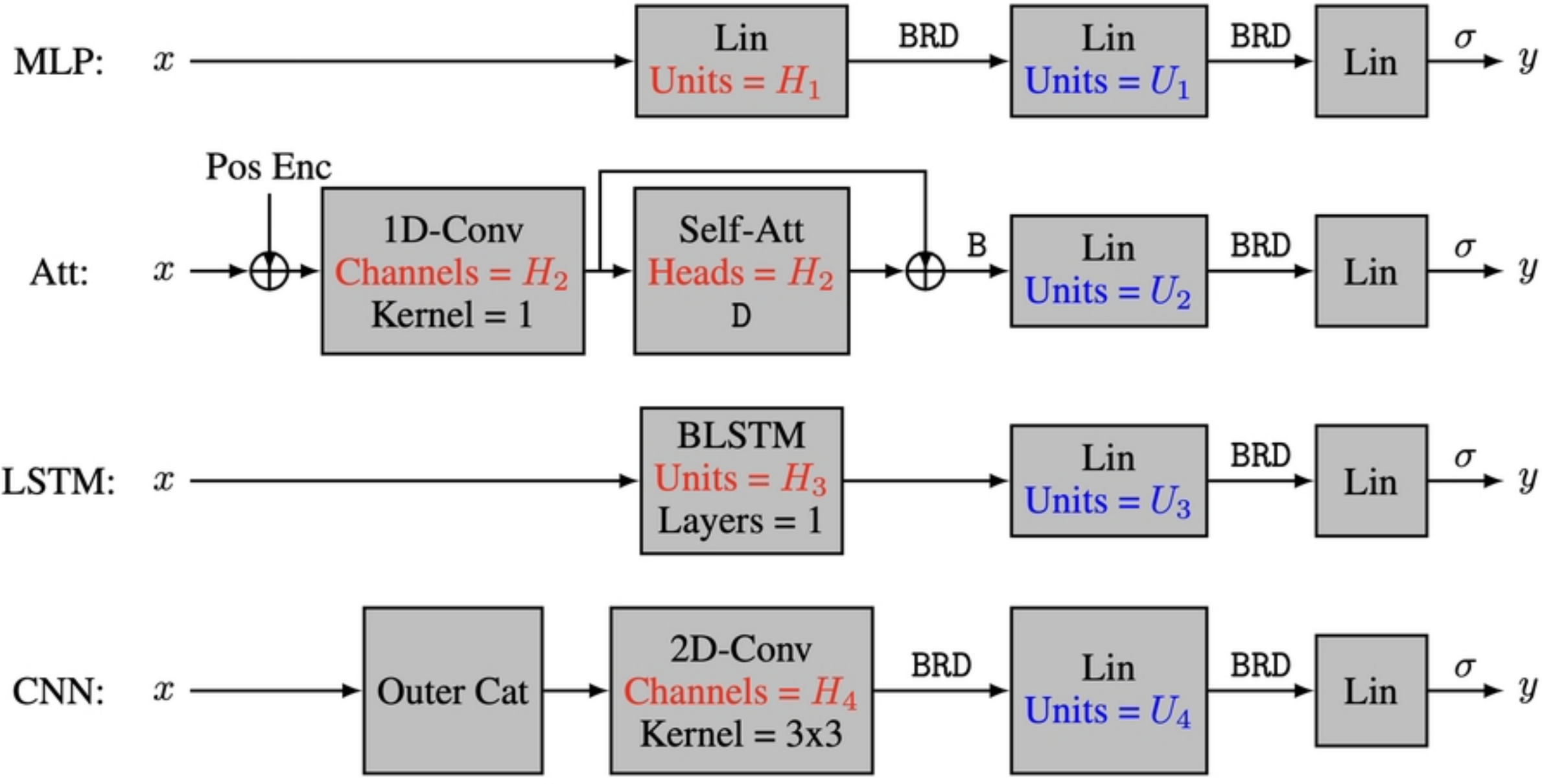

MLP:
x
Lin
Units = H_1
BRD
Lin
Units = U_1
BRD
Lin
σ
y
Pos Enc
Att:
x
1D-Conv
Channels = H_2
Kernel = 1
Self-Att
Heads = H_2
D
B
Lin
Units = U_2
BRD
Lin
σ
y
LSTM:
x
BLSTM
Units = H_3
Layers = 1
Lin
Units = U_3
BRD
Lin
σ
y
CNN:
x
Outer Cat
2D-Conv
Channels = H_4
Kernel = 3x3
BRD
Lin
Units = U_4
BRD
Lin
σ
y

Figure 1

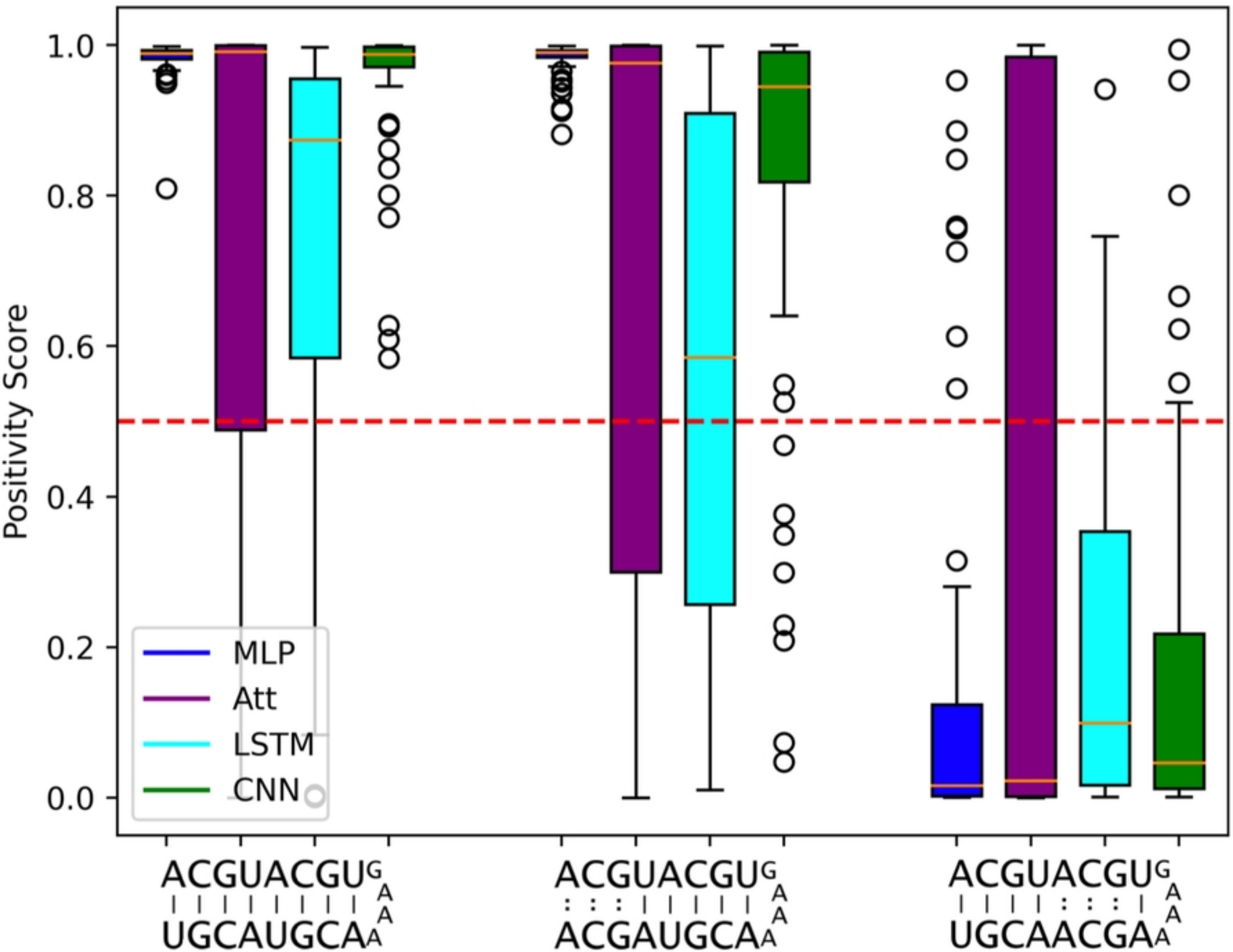

Positivity Score
1.0
0.8
0.6
0.4
0.2
0.0
MLP
Att
LSTM
CNN
ACGUACGUGA
UGCAUGCAA
ACGUACGUGA
ACGAUGCAA
ACGUACGUGA
UGCAACGAA

Figure 6

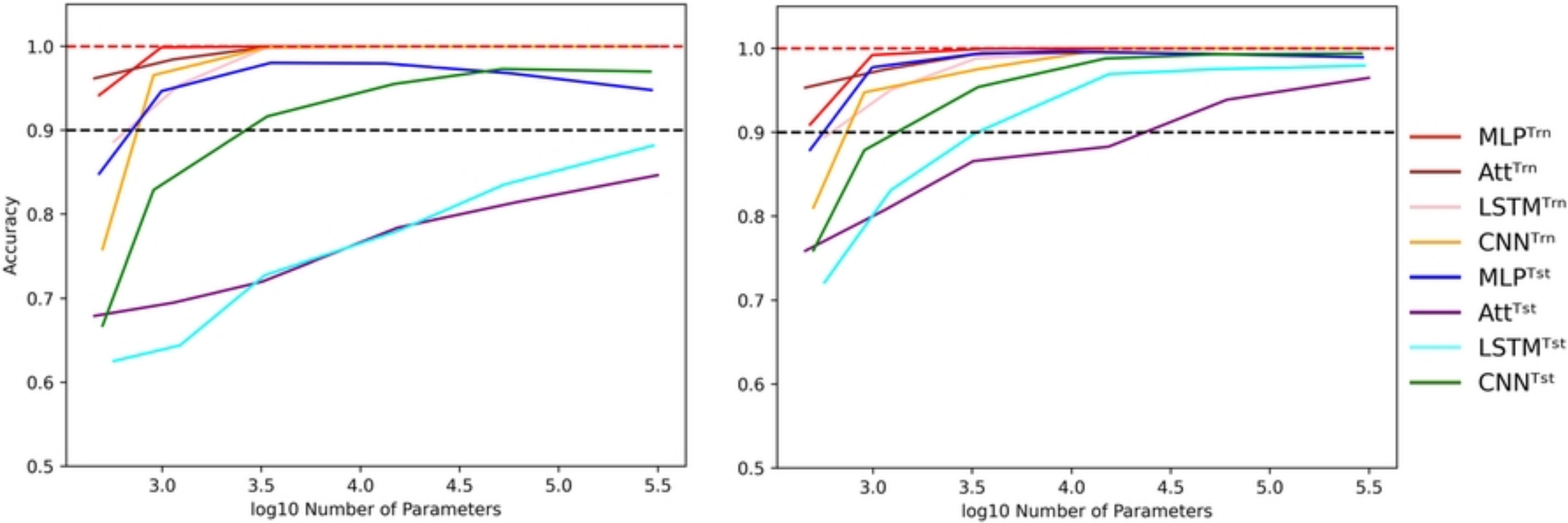

Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
3.0
3.5
4.0
4.5
5.0
5.5
log10 Number of Parameters
MLP^Trn
Att^Trn
LSTM^Trn
CNN^Trn
MLP^Tst
Att^Tst
LSTM^Tst
CNN^Tst

Figure 5

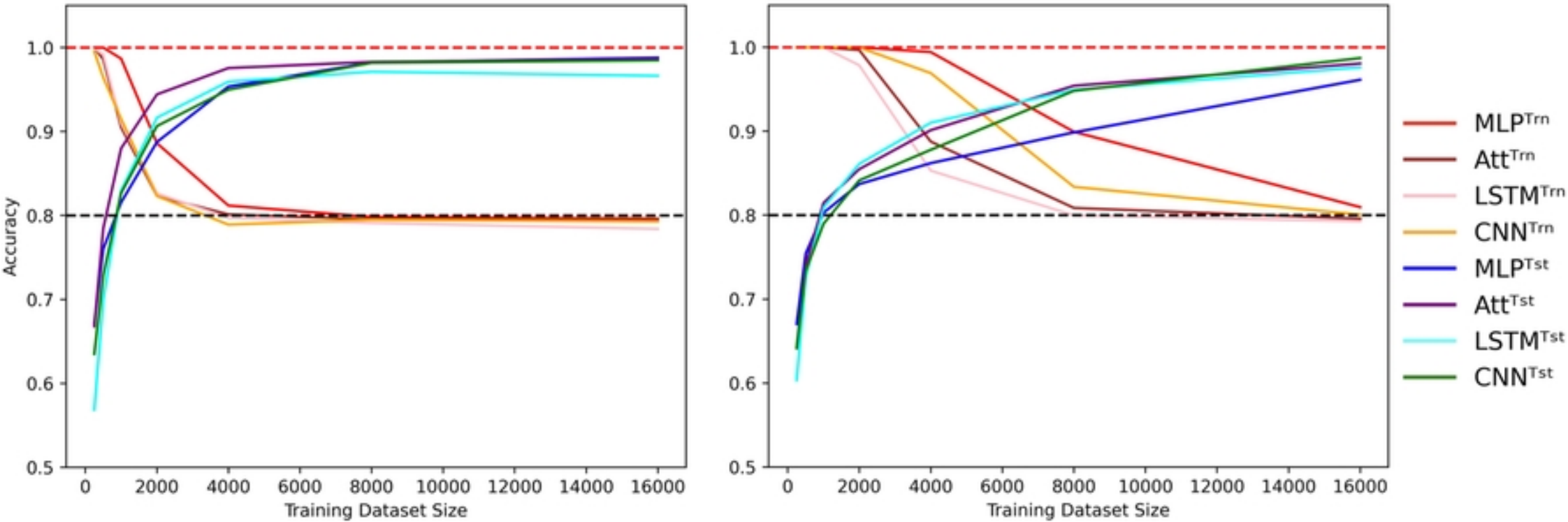

Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0
2000
4000
6000
8000
10000
12000
14000
16000
Training Dataset Size
MLP^Trn
Att^Trn
LSTM^Trn
CNN^Trn
MLP^Tst
Att^Tst
LSTM^Tst
CNN^Tst

Figure 3

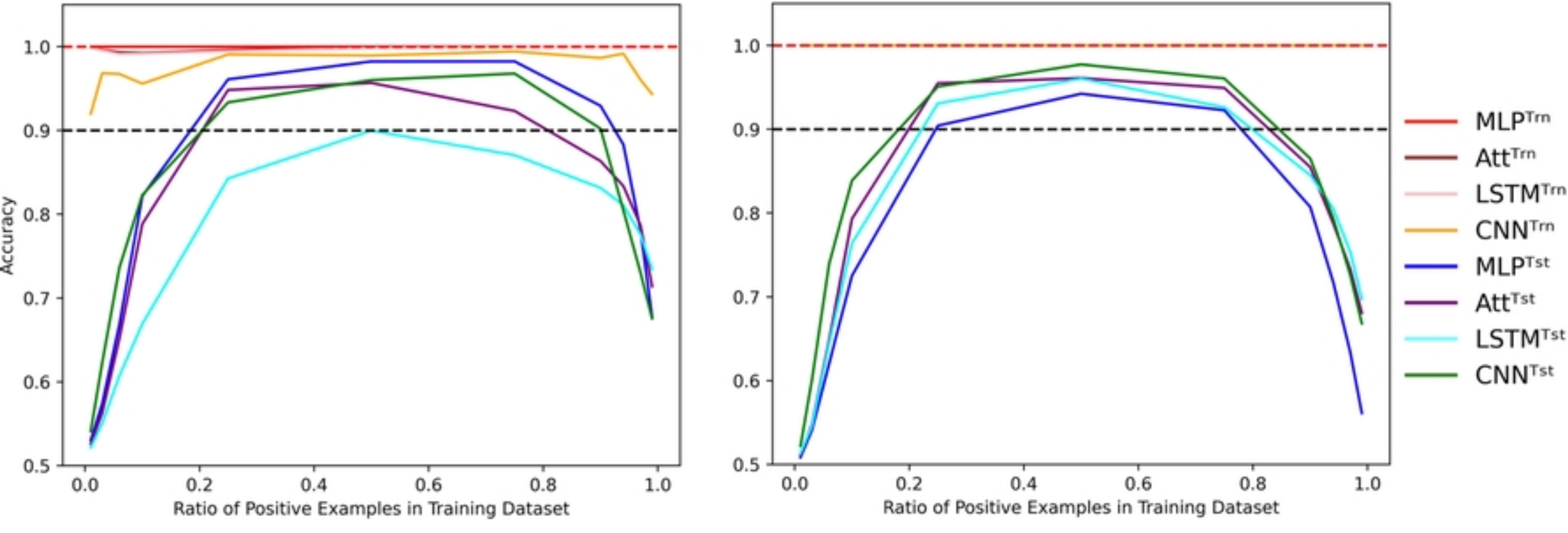

Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Ratio of Positive Examples in Training Dataset
MLP^Trn
Att^Trn
LSTM^Trn
CNN^Trn
MLP^Tst
Att^Tst
LSTM^Tst
CNN^Tst

Figure 7

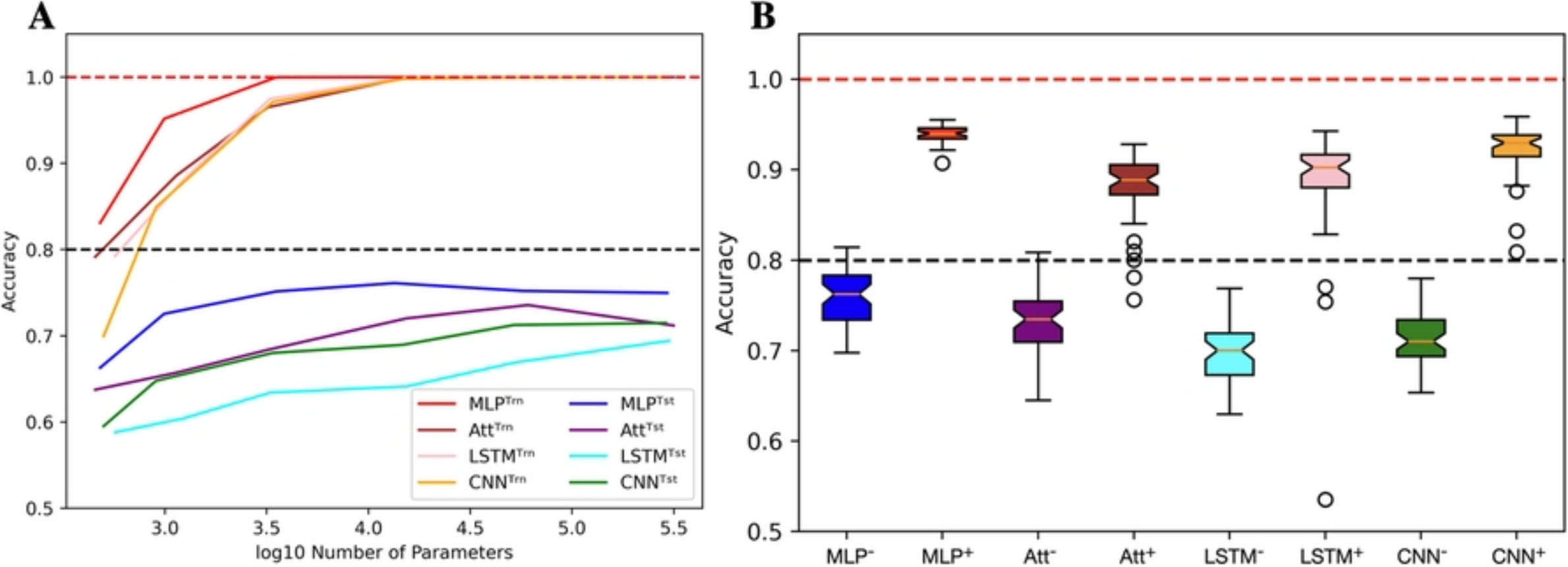

A
B
Accuracy
log10 Number of Parameters
MLP^Trn
Att^Trn
LSTM^Trn
CNN^Trn
MLP^Tst
Att^Tst
LSTM^Tst
CNN^Tst
MLP⁻
MLP⁺
Att⁻
Att⁺
LSTM⁻
LSTM⁺
CNN⁻
CNN⁺

Figure 9

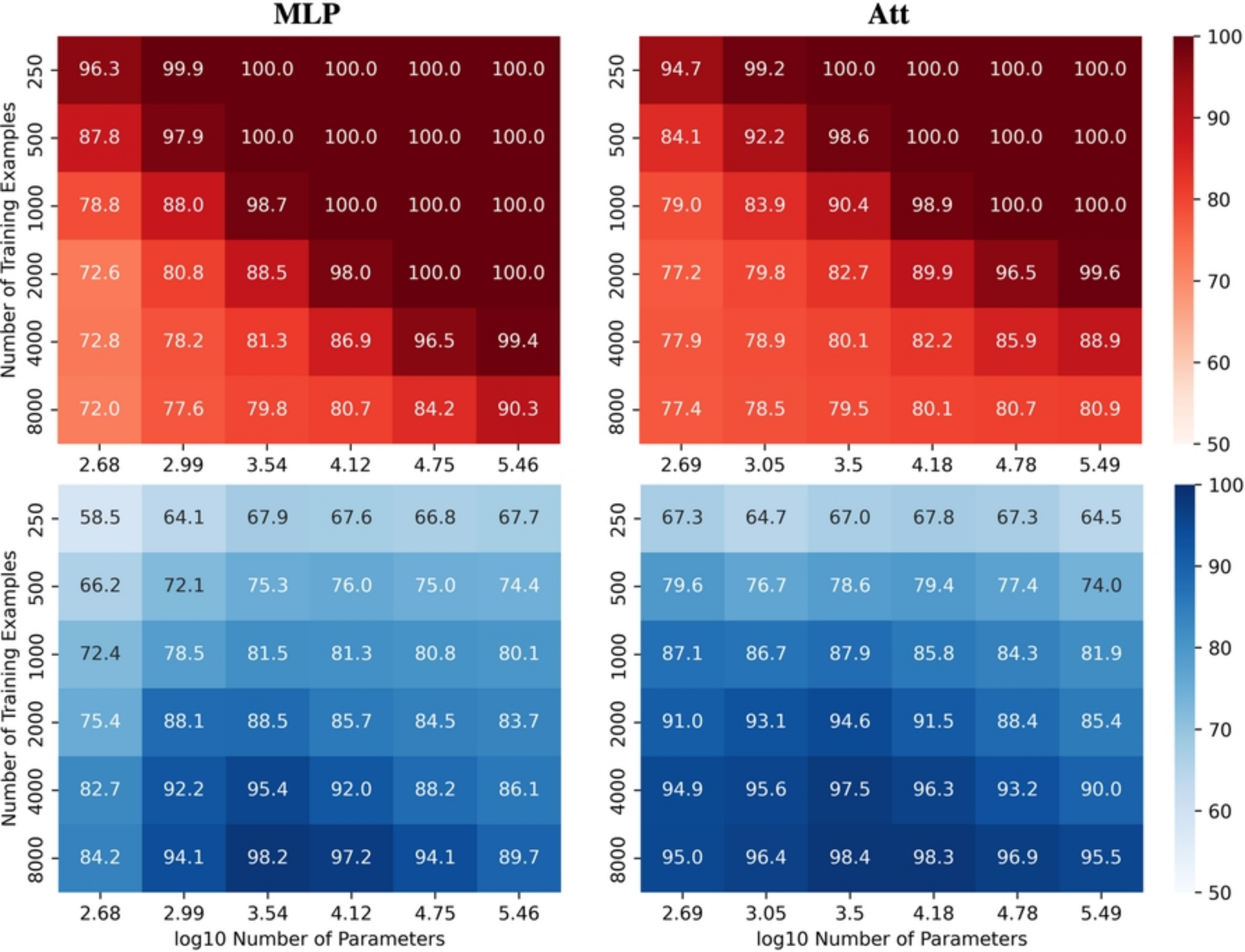

MLP
Att
Number of Training Examples
250
500
1000
2000
4000
8000
96.3 99.9 100.0 100.0 100.0 100.0
87.8 97.9 100.0 100.0 100.0 100.0
78.8 88.0 98.7 100.0 100.0 100.0
72.6 80.8 88.5 98.0 100.0 100.0
72.8 78.2 81.3 86.9 96.5 99.4
72.0 77.6 79.8 80.7 84.2 90.3
94.7 99.2 100.0 100.0 100.0 100.0
84.1 92.2 98.6 100.0 100.0 100.0
79.0 83.9 90.4 98.9 100.0 100.0
77.2 79.8 82.7 89.9 96.5 99.6
77.9 78.9 80.1 82.2 85.9 88.9
77.4 78.5 79.5 80.1 80.7 80.9
2.68 2.99 3.54 4.12 4.75 5.46
2.69 3.05 3.5 4.18 4.78 5.49
100
90
80
70
60
50
58.5 64.1 67.9 67.6 66.8 67.7
66.2 72.1 75.3 76.0 75.0 74.4
72.4 78.5 81.5 81.3 80.8 80.1
75.4 88.1 88.5 85.7 84.5 83.7
82.7 92.2 95.4 92.0 88.2 86.1
84.2 94.1 98.2 97.2 94.1 89.7
67.3 64.7 67.0 67.8 67.3 64.5
79.6 76.7 78.6 79.4 77.4 74.0
87.1 86.7 87.9 85.8 84.3 81.9
91.0 93.1 94.6 91.5 88.4 85.4
94.9 95.6 97.5 96.3 93.2 90.0
95.0 96.4 98.4 98.3 96.9 95.5
log10 Number of Parameters

Figure 2

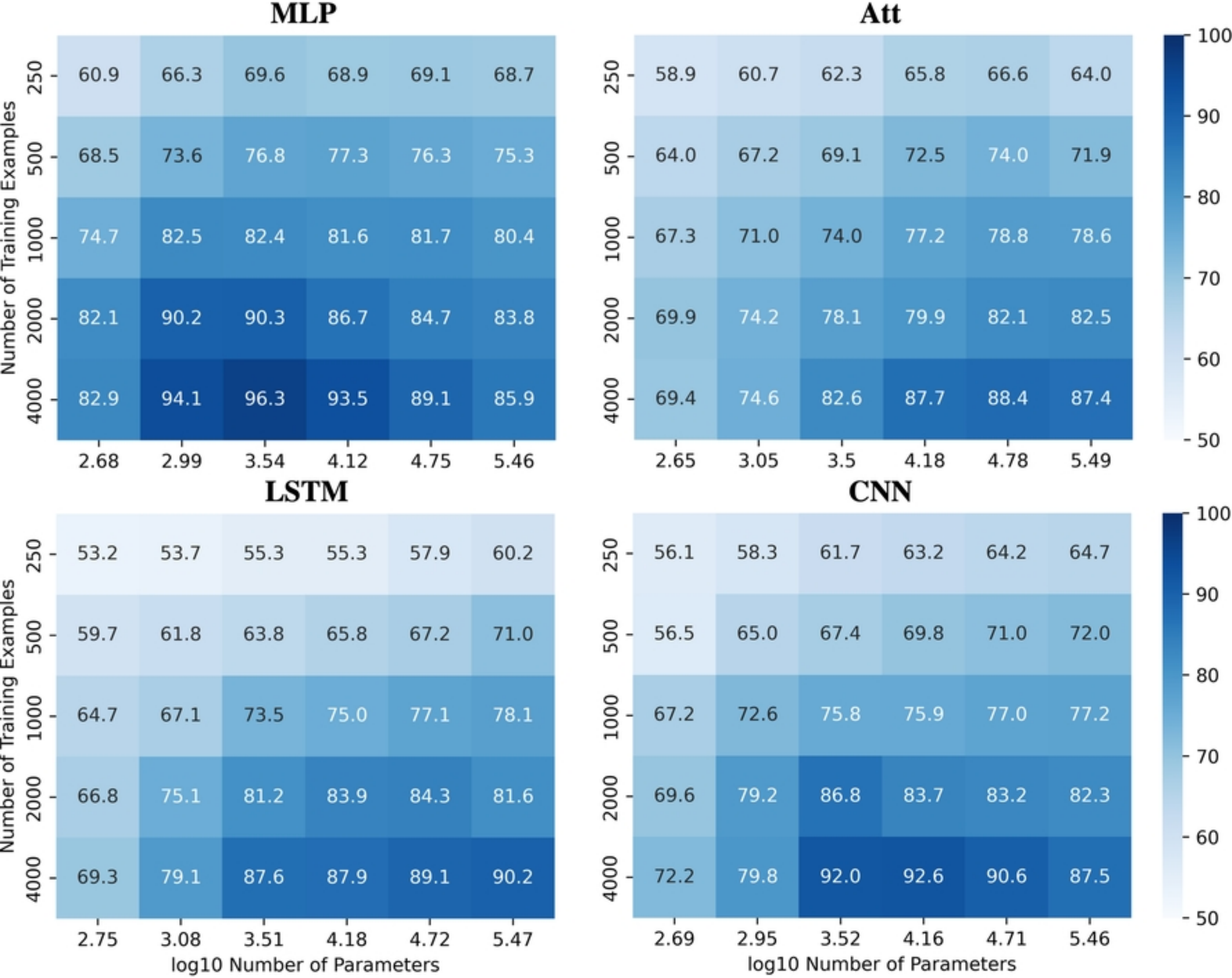

MLP
Att
LSTM
CNN
Number of Training Examples
250
500
1000
2000
4000
60.9 66.3 69.6 68.9 69.1 68.7
68.5 73.6 76.8 77.3 76.3 75.3
74.7 82.5 82.4 81.6 81.7 80.4
82.1 90.2 90.3 86.7 84.7 83.8
82.9 94.1 96.3 93.5 89.1 85.9
2.68 2.99 3.54 4.12 4.75 5.46
58.9 60.7 62.3 65.8 66.6 64.0
64.0 67.2 69.1 72.5 74.0 71.9
67.3 71.0 74.0 77.2 78.8 78.6
69.9 74.2 78.1 79.9 82.1 82.5
69.4 74.6 82.6 87.7 88.4 87.4
2.65 3.05 3.5 4.18 4.78 5.49
53.2 53.7 55.3 55.3 57.9 60.2
59.7 61.8 63.8 65.8 67.2 71.0
64.7 67.1 73.5 75.0 77.1 78.1
66.8 75.1 81.2 83.9 84.3 81.6
69.3 79.1 87.6 87.9 89.1 90.2
2.75 3.08 3.51 4.18 4.72 5.47
56.1 58.3 61.7 63.2 64.2 64.7
56.5 65.0 67.4 69.8 71.0 72.0
67.2 72.6 75.8 75.9 77.0 77.2
69.6 79.2 86.8 83.7 83.2 82.3
72.2 79.8 92.0 92.6 90.6 87.5
2.69 2.95 3.52 4.16 4.71 5.46
log10 Number of Parameters
100
90
80
70
60
50

Figure 8

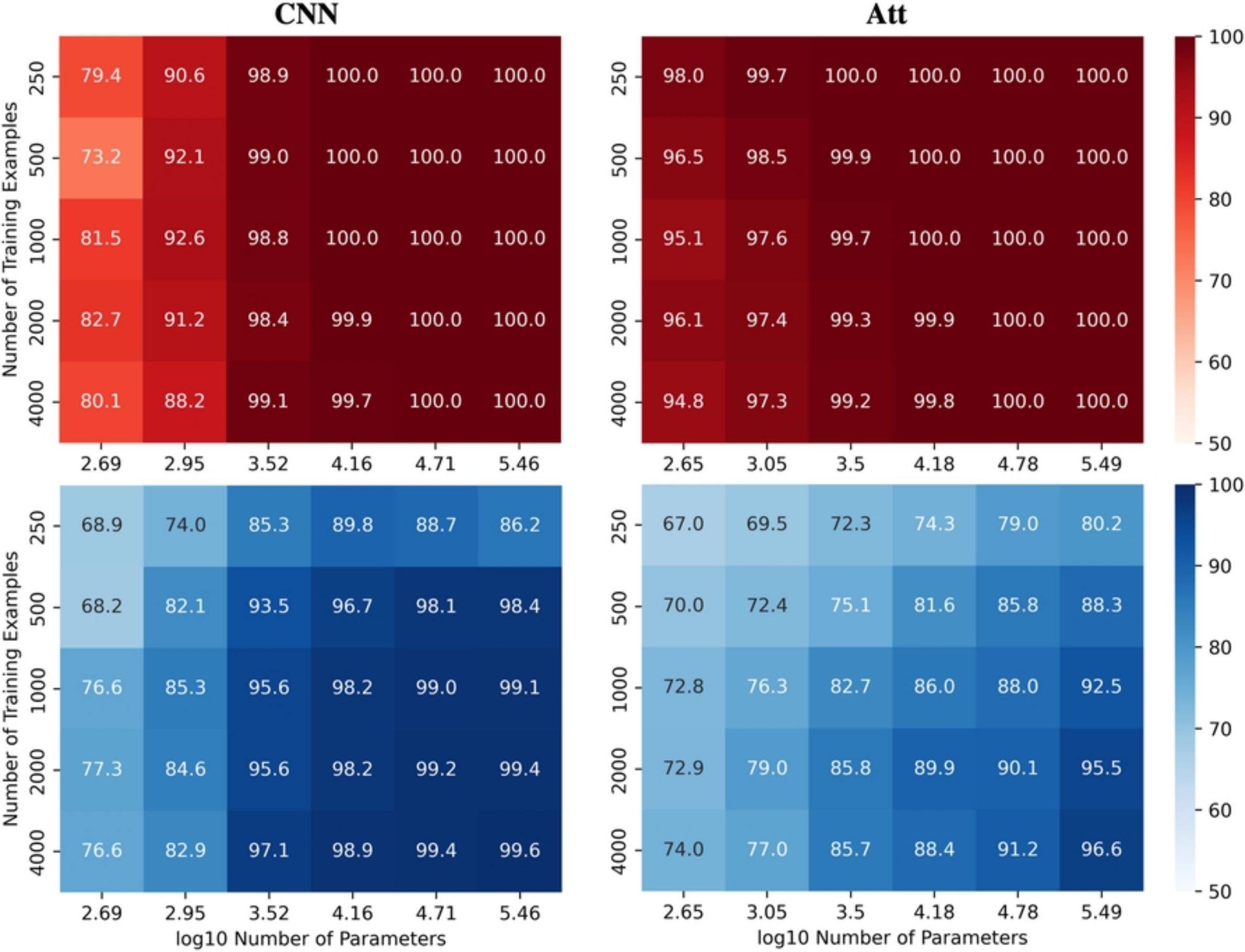

CNN
Att
Number of Training Examples
250
500
1000
2000
4000
79.4 90.6 98.9 100.0 100.0 100.0
73.2 92.1 99.0 100.0 100.0 100.0
81.5 92.6 98.8 100.0 100.0 100.0
82.7 91.2 98.4 99.9 100.0 100.0
80.1 88.2 99.1 99.7 100.0 100.0
98.0 99.7 100.0 100.0 100.0 100.0
96.5 98.5 99.9 100.0 100.0 100.0
95.1 97.6 99.7 100.0 100.0 100.0
96.1 97.4 99.3 99.9 100.0 100.0
94.8 97.3 99.2 99.8 100.0 100.0
68.9 74.0 85.3 89.8 88.7 86.2
68.2 82.1 93.5 96.7 98.1 98.4
76.6 85.3 95.6 98.2 99.0 99.1
77.3 84.6 95.6 98.2 99.2 99.4
76.6 82.9 97.1 98.9 99.4 99.6
67.0 69.5 72.3 74.3 79.0 80.2
70.0 72.4 75.1 81.6 85.8 88.3
72.8 76.3 82.7 86.0 88.0 92.5
72.9 79.0 85.8 89.9 90.1 95.5
74.0 77.0 85.7 88.4 91.2 96.6
2.69 2.95 3.52 4.16 4.71 5.46
2.65 3.05 3.5 4.18 4.78 5.49
log10 Number of Parameters
100
90
80
70
60
50

Figure 4

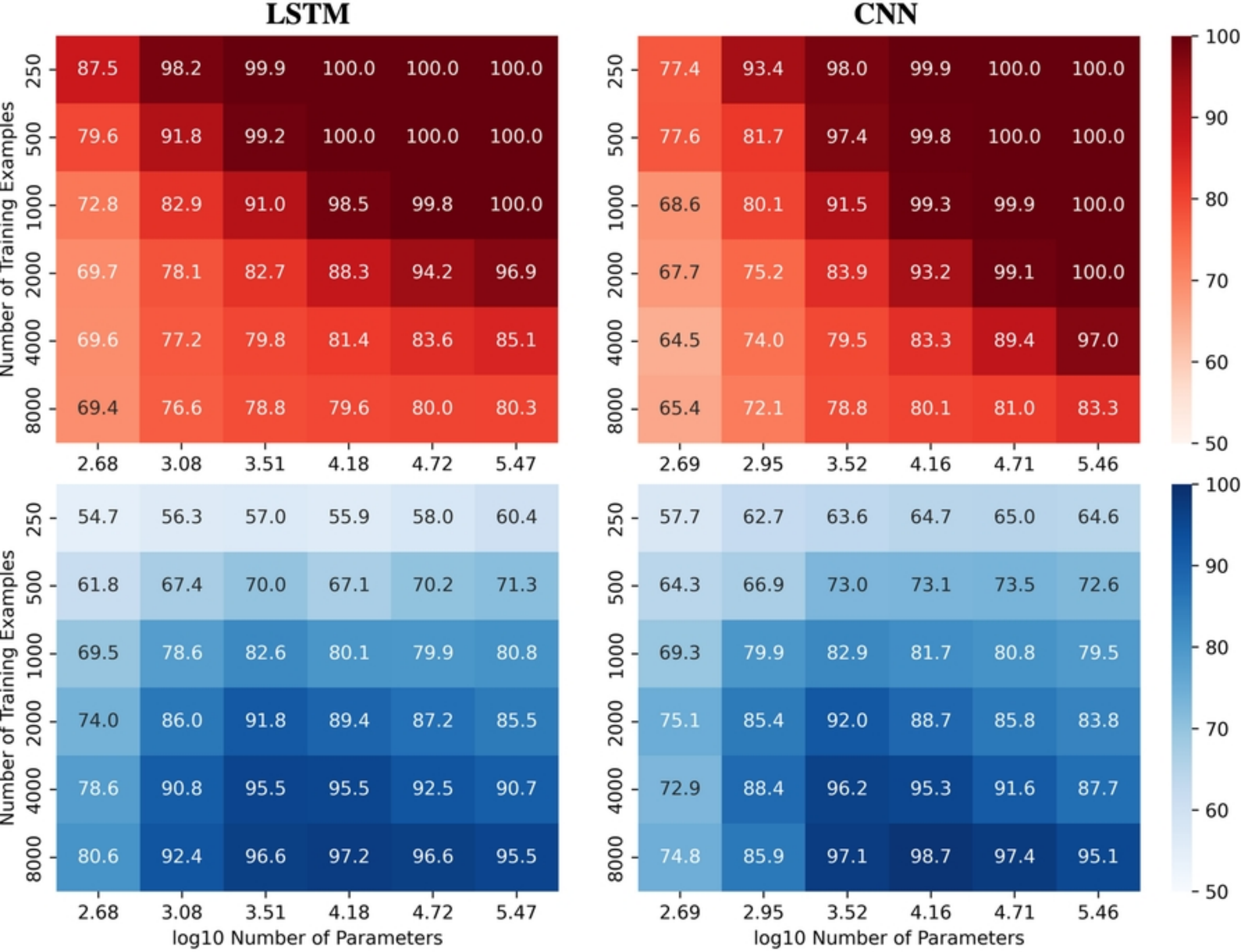

LSTM
CNN
Number of Training Examples
250
500
1000
2000
4000
8000
87.5 98.2 99.9 100.0 100.0 100.0
79.6 91.8 99.2 100.0 100.0 100.0
72.8 82.9 91.0 98.5 99.8 100.0
69.7 78.1 82.7 88.3 94.2 96.9
69.6 77.2 79.8 81.4 83.6 85.1
69.4 76.6 78.8 79.6 80.0 80.3
77.4 93.4 98.0 99.9 100.0 100.0
77.6 81.7 97.4 99.8 100.0 100.0
68.6 80.1 91.5 99.3 99.9 100.0
67.7 75.2 83.9 93.2 99.1 100.0
64.5 74.0 79.5 83.3 89.4 97.0
65.4 72.1 78.8 80.1 81.0 83.3
2.68 3.08 3.51 4.18 4.72 5.47
2.69 2.95 3.52 4.16 4.71 5.46
100
90
80
70
60
50
54.7 56.3 57.0 55.9 58.0 60.4
61.8 67.4 70.0 67.1 70.2 71.3
69.5 78.6 82.6 80.1 79.9 80.8
74.0 86.0 91.8 89.4 87.2 85.5
78.6 90.8 95.5 95.5 92.5 90.7
80.6 92.4 96.6 97.2 96.6 95.5
57.7 62.7 63.6 64.7 65.0 64.6
64.3 66.9 73.0 73.1 73.5 72.6
69.3 79.9 82.9 81.7 80.8 79.5
75.1 85.4 92.0 88.7 85.8 83.8
72.9 88.4 96.2 95.3 91.6 87.7
74.8 85.9 97.1 98.7 97.4 95.1
log10 Number of Parameters

Figure S1

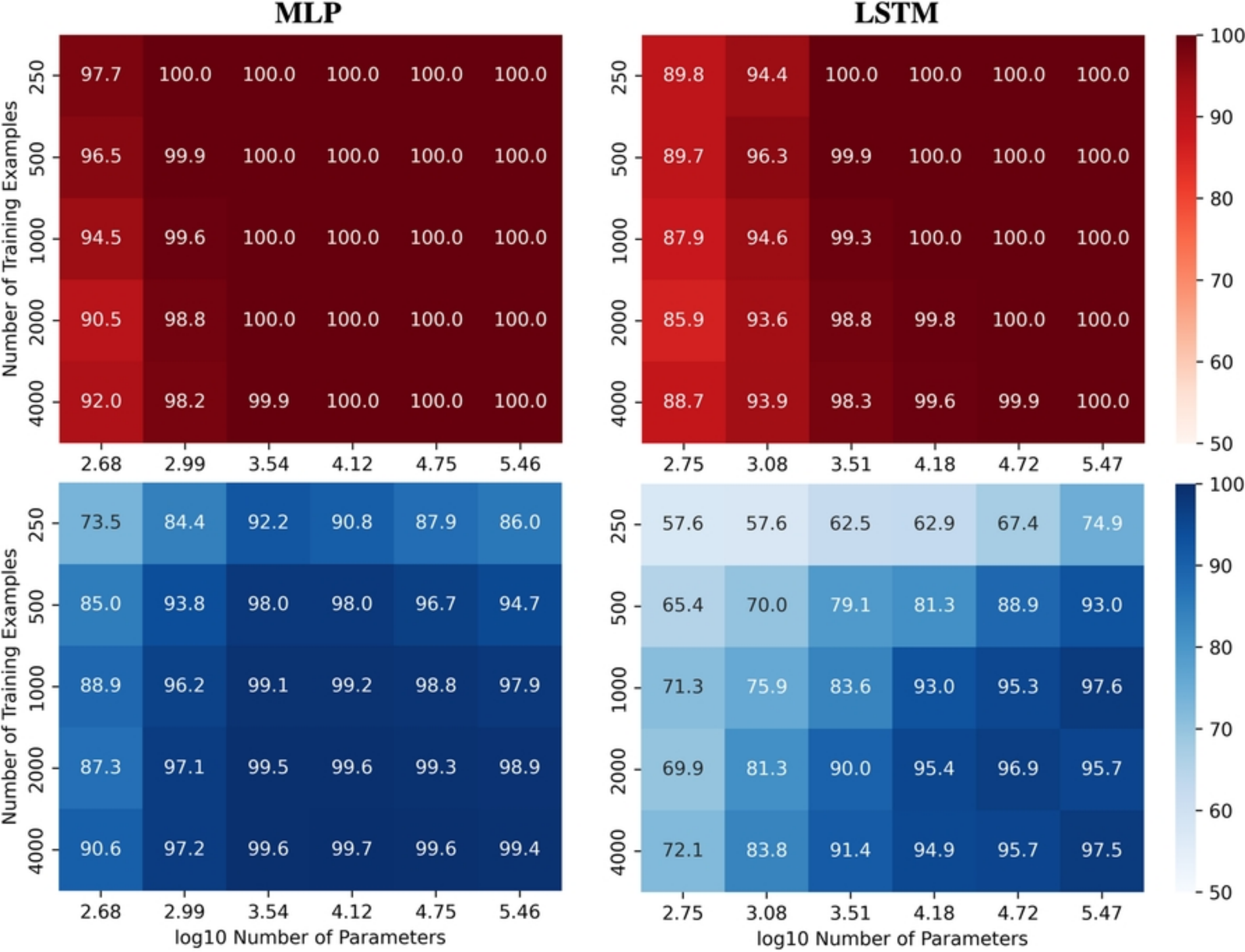
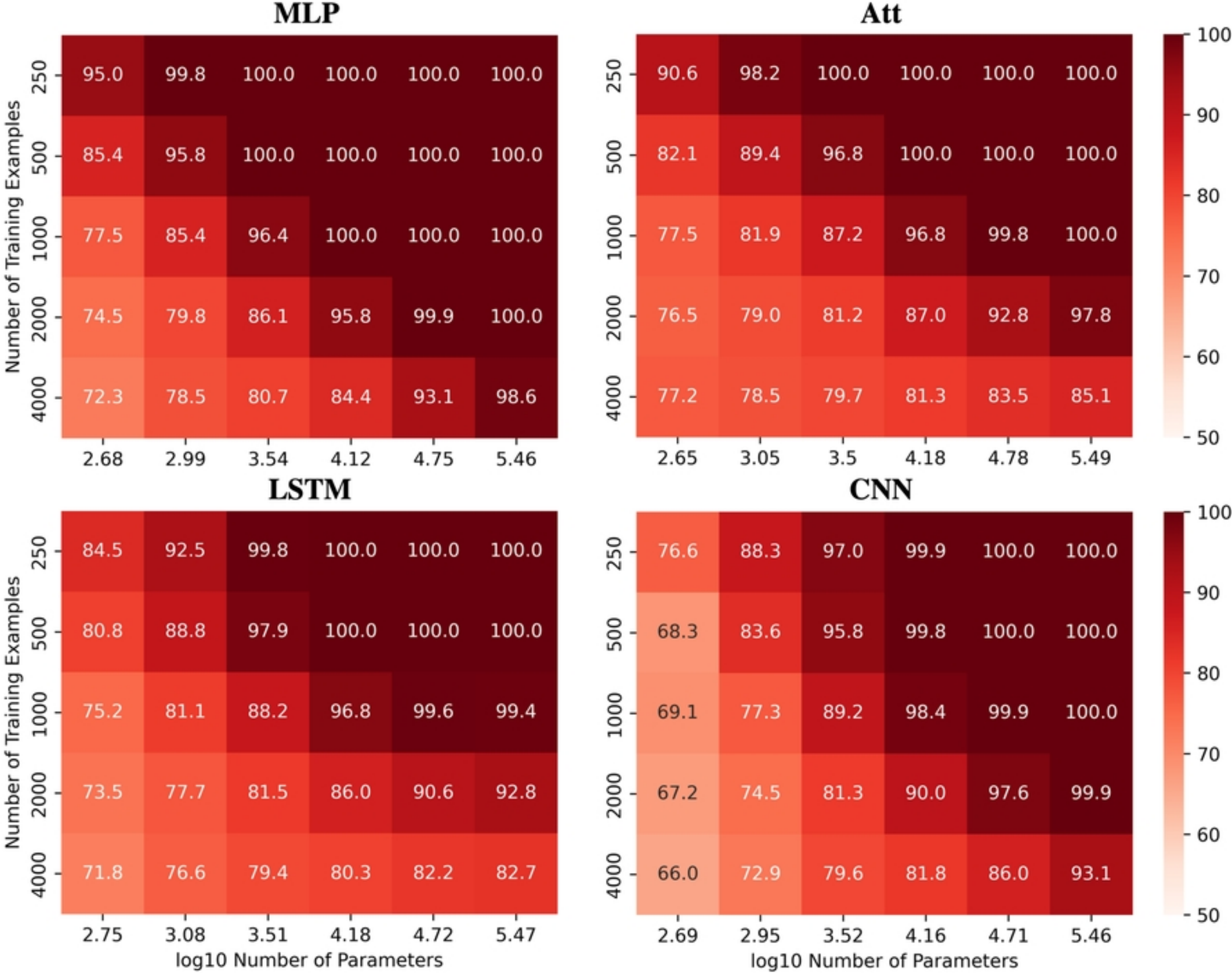

MLP
LSTM
Number of Training Examples
250
500
1000
2000
4000
97.7 100.0 100.0 100.0 100.0 100.0
96.5 99.9 100.0 100.0 100.0 100.0
94.5 99.6 100.0 100.0 100.0 100.0
90.5 98.8 100.0 100.0 100.0 100.0
92.0 98.2 99.9 100.0 100.0 100.0
2.68 2.99 3.54 4.12 4.75 5.46
89.8 94.4 100.0 100.0 100.0 100.0
89.7 96.3 99.9 100.0 100.0 100.0
87.9 94.6 99.3 100.0 100.0 100.0
85.9 93.6 98.8 99.8 100.0 100.0
88.7 93.9 98.3 99.6 99.9 100.0
2.75 3.08 3.51 4.18 4.72 5.47
100
90
80
70
60
50
73.5 84.4 92.2 90.8 87.9 86.0
85.0 93.8 98.0 98.0 96.7 94.7
88.9 96.2 99.1 99.2 98.8 97.9
87.3 97.1 99.5 99.6 99.3 98.9
90.6 97.2 99.6 99.7 99.6 99.4
2.68 2.99 3.54 4.12 4.75 5.46
log10 Number of Parameters
57.6 57.6 62.5 62.9 67.4 74.9
65.4 70.0 79.1 81.3 88.9 93.0
71.3 75.9 83.6 93.0 95.3 97.6
69.9 81.3 90.0 95.4 96.9 95.7
72.1 83.8 91.4 94.9 95.7 97.5
2.75 3.08 3.51 4.18 4.72 5.47
log10 Number of Parameters
100
90
80
70
60
50

Figure S2


MLP
Att
LSTM
CNN
Number of Training Examples
250
500
1000
2000
4000
95.0 99.8 100.0 100.0 100.0 100.0
85.4 95.8 100.0 100.0 100.0 100.0
77.5 85.4 96.4 100.0 100.0 100.0
74.5 79.8 86.1 95.8 99.9 100.0
72.3 78.5 80.7 84.4 93.1 98.6
2.68 2.99 3.54 4.12 4.75 5.46
90.6 98.2 100.0 100.0 100.0 100.0
82.1 89.4 96.8 100.0 100.0 100.0
77.5 81.9 87.2 96.8 99.8 100.0
76.5 79.0 81.2 87.0 92.8 97.8
77.2 78.5 79.7 81.3 83.5 85.1
2.65 3.05 3.5 4.18 4.78 5.49
84.5 92.5 99.8 100.0 100.0 100.0
80.8 88.8 97.9 100.0 100.0 100.0
75.2 81.1 88.2 96.8 99.6 99.4
73.5 77.7 81.5 86.0 90.6 92.8
71.8 76.6 79.4 80.3 82.2 82.7
2.75 3.08 3.51 4.18 4.72 5.47
log10 Number of Parameters
76.6 88.3 97.0 99.9 100.0 100.0
68.3 83.6 95.8 99.8 100.0 100.0
69.1 77.3 89.2 98.4 99.9 100.0
67.2 74.5 81.3 90.0 97.6 99.9
66.0 72.9 79.6 81.8 86.0 93.1
2.69 2.95 3.52 4.16 4.71 5.46
log10 Number of Parameters
100
90
80
70
60
50