Residual Shuffle-Exchange Network for Fast Processing of Long Sequences

Andis Draguns, Emīls Ozoliņš, Agris Šostaks, Matīss Apinis, Kārlis Freivalds

Institute of Mathematics and Computer Science, University of Latvia
{andis.draguns, emils.ozolins, agris.sostaks, matiss.apinis, karlis.freivalds}@lumii.lv

Abstract

Attention is a commonly used mechanism in sequence processing, but it is of $O(n^2)$ complexity which prevents its application to long sequences. The recently introduced neural Shuffle-Exchange network offers a computation-efficient alternative, enabling the modelling of long-range dependencies in $O(n \log n)$ time. The model, however, is quite complex, involving a sophisticated gating mechanism derived from the Gated Recurrent Unit. In this paper, we present a simple and lightweight variant of the Shuffle-Exchange network, which is based on a residual network employing GELU and Layer Normalization. The proposed architecture not only scales to longer sequences but also converges faster and provides better accuracy. It surpasses the Shuffle-Exchange network on the LAMBADA language modelling task and achieves state-of-the-art performance on the MusicNet dataset for music transcription while being efficient in the number of parameters. We show how to combine the improved Shuffle-Exchange network with convolutional layers, establishing it as a useful building block in long sequence processing applications.

Introduction

More and more applications of sequence processing performed by neural networks require dealing with long inputs. A key requirement is to allow modelling of dependencies between distant parts of the sequences. Such long-range dependencies occur in natural language when the meaning of some word depends on other words in the same or previous sentence. There are important cases, e.g., to resolve coreferences, when such distant information may not be disregarded.

In music, dependencies occur on several scales. At the finest scale samples of the waveform correlate to form note pitches, at medium scale neighbouring notes relate to each other by forming melodies and chord progressions, at coarse scale common melodies reappear throughout the entire piece creating a coherent musical form (Thickstun, Harchaoui, and Kakade 2017; Huang et al. 2019). Dealing with such dependencies requires processing very long sequences (several pages of text or the entire musical composition) in a manner that aggregates information from their distant parts. Especially challenging are approaches that work directly on the raw audio waveform.

The ability to combine distant information is even more important for algorithmic tasks where each output symbol typically depends on every input symbol. The goal of algorithm induction is to derive an algorithm from input-output examples which are often given as sequences. Algorithmic tasks are especially challenging due to the need for processing sequences of unlimited length. Also, generalization plays an important role since training is often performed on short sequences but testing on long ones.

The commonly used attention mechanism (for example, in Transformers) can deal with such long-range dependencies but its time and space complexity is quadratic depending on the sequence length, therefore, is not an attractive choice for long sequences. The attention mechanism's complexity also makes it slower at inference time, making it less suitable for tasks with strict latency requirements, such as realtime sound processing. The recently introduced neural Shuffle-Exchange networks allow modelling of long-range dependencies in sequences in $O(n \log n)$ time (Freivalds, Ozoliņš, and Šostaks 2019). The idea is very promising and offers a computation-efficient alternative to the attention mechanism. However, the model is quite complex, involving a sophisticated gating mechanism derived from the Gated Recurrent Unit (Cho et al. 2014).

In this paper, we present a much simpler and faster version of the neural Shuffle-Exchange network which is based on the residual network idea employing Gaussian Error Linear Units (Hendrycks and Gimpel 2016) and Layer Normalization (Ba, Kiros, and Hinton 2016) instead of gates.

We empirically validate our improved model on algorithmic tasks, LAMBADA question answering and multi-instrument musical note recognition (MusicNet dataset). It surpasses the original Shuffle-Exchange network by 2.1% on the LAMBADA language modelling task and achieves state-of-the-art 78.02% average precision score on Music-Net.

We introduce a modification where we prepend our proposed model with strided convolutions to increase the speed and applicability to long sequences even more. This change enables processing a sequence of length 2M symbols in only 3.97 seconds.

Our main contributions are:

- We propose a much simpler and faster Switch Unit – the core part of the Shuffle-Exchange network. This improve-

ment leads to higher accuracy and scaling to longer sequences. It also makes the model approximately 4 times faster to train and 2 times faster on inference.

- We surpass the previous state-of-the-art on MusicNet while being efficient in the number of parameters. Our proposed improvements to the architecture enable this state-of-the-art achieving model to run inference on a single GPU fast enough to be suitable for realtime audio processing.

Related Work

The attention mechanism (Bahdanau, Cho, and Bengio 2014) has become a standard choice in numerous neural models, including Transformer (Vaswani et al. 2017) and BERT (Devlin et al. 2018) which achieved state-of-the-art accuracy in NLP and related tasks. However, the complexity of the attention mechanism is quadratic depending on the input length and does not scale to long sequences.

One way to overcome the complexity of attention is cutting the sequence into short segments and using attention only within the segment boundaries (Al-Rfou et al. 2018). Various sparse attention mechanisms have been proposed to deal with the quadratic complexity of dense attention by attending only to a small predetermined subset of locations (Beltagy, Peters, and Cohan 2020; Zaheer et al. 2020). Reformer (Kitaev, Kaiser, and Levskaya 2020) uses locality-sensitive hashing to approximate attention in time O($n \log n$). Linformer (Wang et al. 2020) uses a linear complexity approximation to the original attention by creating a low-rank factorization of the attention matrix.

Another option for processing long sequences is using convolutional architectures (Gehring et al. 2017). However, convolutions are inherently local — the value of a particular neuron depends on a small neighbourhood of the previous layer. One way to capture long-range structure is to increase the receptive field of convolution by using dilated (atrous) convolution, where the convolution mask is spread out at regular spatial intervals. Dilated architectures have achieved great success in image segmentation (Yu and Koltun 2015) and audio generation (van den Oord et al. 2016).

An important use of sequence processing models is in learning algorithmic tasks (see Kant (2018) for a good overview) where the way how memory is accessed is crucial. Neural GPU (Kaiser and Sutskever 2015) utilizes active memory (Kaiser and Bengio 2016) where computation is coupled with memory access. Freivalds and Liepins (2018) proposes DNGPU where the flow of information in the Neural GPU is facilitated by introducing diagonal gates that improves training and generalization but does not address the performance problem caused by many layers.

For music transcription tasks convolutional architectures are common. See (Benetos et al. 2019) for a good overview. Trabelsi et al. (2018) achieves notable performance on MusicNet by using a convolutional network based on complex numbers. Yang et al. (2020) recently proposed a Transformer network that employs the Fourier transform of the waveform in the complex domain.

Neural Shuffle-Exchange Networks

Neural Shuffle-Exchange network (Freivalds, Ozoliņš, and Šostaks 2019) has been recently proposed as an efficient alternative to the attention mechanism that allows modelling of long-range dependencies in sequences in O($n \log n$) time. The neural Shuffle-Exchange network is the neural adaption of the Shuffle-Exchange and the Beneš networks which are well-known from packet routing tasks in computer networks. These networks consist of interleaved shuffle and switch layers. The shuffle layer permutes the signals. The switch layer consists of switches. Each switch is configured to either swap two adjacent signals or leave them unchanged. The neural Shuffle-Exchange network adapts the structure of Beneš network and replaces each switch with a Switch Unit – a learnable 2-to-2 function. See Fig. 1 for an example of a neural Shuffle-Exchange network routing signals from 8 input sequence addresses to 8 output sequence addresses.

The input to the neural Shuffle-Exchange network is a sequence of length $n = 2^k, k \in \mathbb{N}$, and each element of the sequence is a vector with m dimensions. The input sequence is padded to the nearest power of two. The first layer of the model is a Switch Layer, which consists of $n/2$ Switch Units (SU). The input sequence is split into non-overlapping pairs of adjacent elements, and each pair is processed by a Switch Unit. Each Switch Unit is a neural network that computes a 2-to-2 function from a pair of elements. The second layer of the network is a Shuffle Layer, which permutes the inputs according to the perfect shuffle permutation. Perfect shuffle permutation maps each input address to an address that is circularly bit-shifted to the left (e.g. 101 to 011).

Note that all Shuffle Layers are identical and have no learnable parameters. It is also worth noting that both types of layers leave the dimensions of the input unchanged. The third layer of the network is the same as the first layer – a Switch Layer. The layers keep alternating between Shuffle Layers and Switch Layers until there are a total of $\log(n) - 1$ from each type of layer. This arrangement of layers consti-
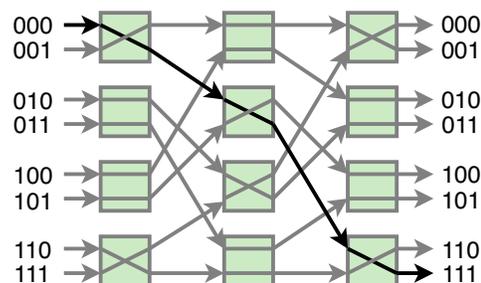

Figure 1: Shuffle-Exchange network routing signals from 8 input sequence addresses (sequence element locations) from the left to 8 output sequence addresses to the right. Each green block is a Switch Unit that takes two input elements and either swaps or leaves them unchanged. A column of Switch Units forms a Switch Layer. The arrows between two Switch Layers represent a Shuffle Layer that permutes the elements. In this figure, the switches are configured to route an element from address 000 to 111.
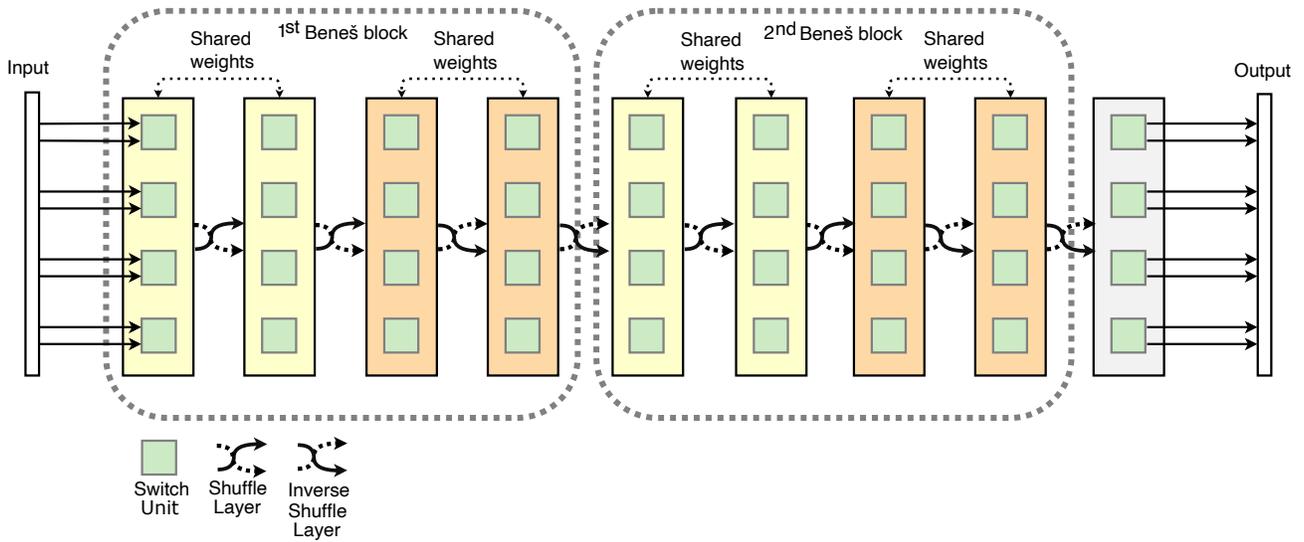
Figure 2: Residual Shuffle-Exchange network with two Beneš blocks and eight inputs. All learnable parameters are within the Switch Units. The rest of the network is fixed and used for routing information. Any number of Beneš blocks can be added to increase the depth of the model.

tutes the Shuffle-Exchange network.

The Shuffle-Exchange network is followed by a reversed Shuffle-Exchange network. The only difference between these two is that in the reversed Shuffle-Exchange network, all Shuffle Layers are replaced by inverse Shuffle Layers. Inverse Shuffle Layer permutes inputs like the regular Shuffle Layer, but the bit shift direction is to the right (e.g. 011 to 101). This combination of regular Shuffle-Exchange network followed by a reversed Shuffle-Exchange network forms a Beneš block – a building block of the neural Shuffle-Exchange network. It has been shown (Dally and Towles 2004) that such a Beneš block can connect any input to any output for each input-output simultaneously. Therefore, the neural Shuffle-Exchange network has a 'receptive field' of the size of the whole sequence, and it has no bottleneck. These properties hold for dense attention but have not been shown for many sparse attention and dilated convolutional architectures.

Multiple Beneš blocks can be stacked one after another to increase the depth of the model. After the last Beneš block, a final Switch Layer is added to complete the model. Within each Beneš block the weights are shared for each Switch Unit in the Shuffle-Exchange network and each Switch Unit in the reversed Shuffle-Exchange network (see Fig. 2). Such weight sharing enables generalization on algorithmic tasks because otherwise there would be no straightforward way of scaling up the model for sequences that are longer than the ones observed during training. No significant decrease in accuracy is observed on other tasks as a result of this weight sharing scheme.

At the heart of the neural Shuffle-Exchange network is the Switch Unit. The definition of the original Switch Unit from the neural Shuffle-Exchange network is:

$$s = [s^1, s^2]$$
$$r^1 = \sigma(W_r^1 s + B_r^1)$$
$$r^2 = \sigma(W_r^2 s + B_r^2)$$
$$c^1 = \tanh(W_c 1(r^1 \odot s) + B_c^1)$$
$$c^2 = \tanh(W_c 2(r^2 \odot s) + B_c^2)$$
$$u = \sigma(W_u s + B_u)$$
$$\tilde{s} = \text{swapHalf}(s^1, s^2)$$
$$[s_o^1, s_o^2] = u \odot \tilde{s} + (1 - u) \odot [c^1, c^2]$$

For a more detailed description of how this Switch Unit works, see Freivalds, Ozoliņš, and Šostaks (2019). This formulation of the Switch Unit uses sophisticated gating mechanisms similar to Gated Recurrent Unit (Cho et al. 2014) and is quite complex. Because apart from Switch Units, the network consists of only fixed non-trainable permutations, the choice of Switch Unit is critical to the overall performance of the network. Some alternatives to the Switch Unit have been explored, but so far the simpler architectures have led to a decrease in performance (Freivalds, Ozoliņš, and Šostaks 2019).

Residual Shuffle-Exchange Network

We propose the Residual Shuffle-Exchange network (RSE), which keeps the structure from the neural Shuffle-Exchange network but replaces the Switch Unit with our Residual Switch Unit (RSU). RSU is based on a residual network and employs Gaussian Error Linear Unit (GELU) (Hendrycks and Gimpel 2016) and Layer Normalization. The unit's design is similar to the feed-forward block in the Transformer.

RSU takes as an input two vectors $[i_1, i_2]$ and produces two vectors $[o_1, o_2]$ as an output. Each of these vectors is of

size m, where m is the number of feature maps.

RSU consists of two linear transformations on the feature dimension. The first linear transformation is followed by Layer Normalization (LayerNorm) without output bias and gain (Xu et al. 2019) and then by GELU. By default, we use a 2x larger hidden layer size than the input of the first layer, which is a good compromise of speed and accuracy. A second linear transformation is applied after GELU. The RSU is defined as follows:

$$i = [i_1, i_2]$$
$$g = \text{GELU}(\text{LayerNorm}(Zi))$$
$$c = Wg + B$$
$$[o_1, o_2] = \sigma(S) \odot i + h \odot c$$

In the above equations, Z, W are weight matrices of size $2m \times 4m$ and $4m \times 2m$, respectively, S is vector of size $2m$ and B is a bias vector $-$ all of those are learnable parameters; h is scalar value, $\odot$ denotes element-wise vector multiplication and σ is the sigmoid function. For a visualization of the Residual Switch Unit, see Appendix A.

In this unit, we use a residual connection that is scaled by the learnable parameter S, which is restricted to the range [0,1] by the sigmoid function. Additionally, we scale the new value c coming out of the last linear transformation by a constant h. We initialize S and h such that the signal travelling through the network keeps its expected amplitude at 0.25 under the assumption of the normal distribution (which is observed in practice). To have that, we initialize S as $\sigma^{-1}(r)$ and h as $\sqrt{1 - r^2} * 0.25$ where r is an experimentally chosen constant close to 1. As r approaches 1, RSU and RSE both start to behave more like identity functions. We use $r = 0.9$, which works well. The signal amplitude after LayerNorm is 1, the weight matrix W is initialized to keep this amplitude. If the amplitude of the input is 0.25, then the expected amplitude at the output is also 0.25, which is a good range for the softmax loss. During training, the network is free to adjust these amplitudes, but this initialization provides stable convergence even for deep networks.

Besides being simpler, the improved design of the RSU allows not using skip connections between Beneš blocks that were needed in the original SE network to ensure stable convergence.

Prepending Convolutions

There are tasks, e.g. the MusicNet task, where there is a large mismatch of information content between input and the Residual Shuffle-Exchange network – each input unit contains one sample, but Residual Shuffle-Exchange network requires a large number of feature maps to work well. Encoding just one number into many feature maps is wasteful. For such tasks, we prepend the Residual Shuffle-Exchange network with several strided convolutions to increase the number of feature maps and reduce the sequence length. We use convolutions with stride 2 and apply LayerNorm and GELU after each convolution like in the RSU. Before the result is passed to the Residual Shuffle-Exchange network, a linear transformation is applied. An illustration of a concrete example model with two prepended convolution layers can be found in Appendix C.

Prepending convolutions shortens the input to the RSE network and speeds up processing. The obtained accuracy for the MusicNet is roughly the same. Note that this approach leads to a shorter output than the input. It may be necessary to append transposed convolution layers at the end of the network to upsample the signal back to its original length. For the MusicNet task, upsampling is not necessary since we utilize only a few elements of the output sequence.

Evaluation

We have implemented the proposed architecture in Tensor-Flow. The code is at https://github.com/LUMII-Syslab/RSE. All models are trained on a single Nvidia RTX 2080 Ti (11GB) GPU with RAdam optimizer (Liu et al. 2019)

Our models were hand-tuned based on a coarse grid search of the parameters. Further increasing the model size leads to overfitting or negligible accuracy increase. For the other models, we use the hyperparameters that their authors have reported achieving the highest accuracy.

Algorithmic Tasks

Let us evaluate how well the Residual Shuffle-Exchange (RSE) network performs on algorithmic tasks in comparison with the neural Shuffle-Exchange (SE) (Freivalds, Ozoliņš, and Šostaks 2019). The goal is to infer $O(n \log n)$ time algorithms purely from input-output examples. In these tasks, a single bit change in the input can lead to a completely different output. Algorithmic tasks are good benchmarks to evaluate the model's ability to develop a rich set of long-term dependencies.

We consider long binary addition, long binary multiplication and sorting, which are common benchmark tasks in several papers including (Freivalds, Ozoliņš, and Šostaks 2019; Kalchbrenner, Danihelka, and Graves 2015; Zaremba and Sutskever 2015; Zaremba et al. 2016; Joulin and Mikolov 2015; Grefenstette et al. 2015; Kaiser and Sutskever 2015; Freivalds and Liepins 2018; Dehghani et al. 2018).

The model for evaluation consists of an embedding layer where each symbol of the input is mapped to a vector of length m, one or two Beneš blocks and the output layer which performs a linear transformation to the required number of classes with a softmax cross-entropy loss for each symbol independently. We use the RSE model having one Beneš block for addition and sorting tasks, two blocks for the multiplication task and $m = 192$ feature maps.

We use dataset generators and curriculum learning from the article introducing neural Shuffle-Exchange networks (Freivalds, Ozoliņš, and Šostaks 2019). For training, we instantiate several models for sequence lengths (powers of 2) from 8 to 64 sharing the same weights and train each example on the smallest instance it fits. We pad the sequence up to the required length with zeroes. Figure 3 shows the testing accuracy on sequences of length 64 vs training step. We can see that on the multiplication task the proposed model trains much faster than SE, reaching near-zero error in about 20K steps vs 200K steps for the SE. For addition and sorting tasks, both models perform similarly.

Figure 3: Test error depending on the training step for Residual Shuffle-Exchange (RSE) and Shuffle-Exchange (SE) networks on algorithmic tasks of length 64.
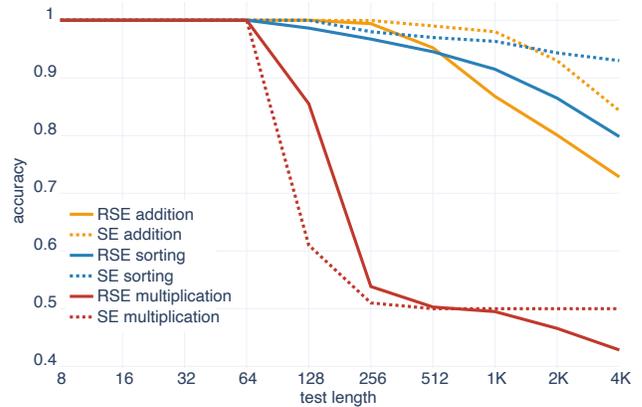

Figure 4: Test accuracy depending on the length for the generalization of Residual Shuffle-Exchange (RSE) and Shuffle-Exchange (SE) to longer sequences on algorithmic tasks.

We have compared the generalization performance of both models, see Fig. 4. We train both models on length up to 64 and evaluate on length up to 4K. On addition and sorting tasks, the proposed RSE model generalizes very well to length 256 but loses slightly to SE on longer sequences. For the multiplication task RSE model generalizes reasonably well to twice as long sequences but not more, where the old model does not generalize even this much. We have compared our model to the DNGPU – the improved Neural GPU (Freivalds and Liepins 2018). Our model outperforms it on all tasks except multiplication. More detailed comparison of DNGPU and RSE generalization performance can be found in Appendix B.

LAMBADA Question Answering

The goal of the LAMBADA task is to predict a given target word from its broad context (on average, 4.6 sentences collected from novels). The sentences in the LAMBADA dataset (Paperno et al. 2016) are specially selected such that giving the right answer requires examining the whole passage. In 81% cases of the test set the target word can be found in the text, and we follow a common strategy (Chu et al. 2017; Dehghani et al. 2018) to choose the target word as one from the text. The answer will be wrong in the remaining cases, so the achieved accuracy will not exceed 81%. Choosing a random word from the passage gives 1.6% test accuracy (Paperno et al. 2016).

We instantiate the model for input length 256 (all test and train examples fit into this length) and pad the input sequence to that length by placing the sequence at a random position and adding zeros on both ends. Randomized padding improves test accuracy. We use a pretrained fastText 1M English word embedding (Mikolov et al. 2018) for the input words. The embedding layer is followed by 2 Beneš blocks with 384 feature maps. To perform the answer selection as a word from the text, each symbol of the output is linearly mapped to a single scalar and we use softmax loss

over the obtained sequence to select the position of the answer word.

In Table 1, we show the test accuracy and the number of learnable parameters of our model in the context of results reported in previous works. The Residual Shuffle-Exchange network scores better than SE by 2.1% while using 3x less learnable parameters (Freivalds, Ozoliņš, and Šostaks 2019). Current state-of-the-art model GPT-3 surpasses our model, achieving 86.4% accuracy while using about 16000 times more learnable parameters and pretraining on a huge dataset (Brown et al. 2020). The performance of GPT-3 is comparable to human performance on this task (Chu et al. 2017). Our model scores lower than Universal Transformer by 1.66% (Dehghani et al. 2018) but uses about 14 times fewer parameters. Our model also scores by 5.34% higher than the Gated-Attention reader (Chu et al. 2017).

In Fig 5, we compare the training and evaluation time of Residual Shuffle-Exchange (RSE), neural Shuffle-Exchange (SE) and Universal Transformer (UT) networks using configurations that reach their best test accuracy. We use the official Universal Transformer and neural Shuffle-Exchange implementations and measure the time for one training and evaluation step on a single sequence. For the Universal Transformer, we use its base configuration with 152M learn-

Model	Parameters (M)	Acc (%)
Random word	-	1.6
Gated-Attention Reader	unknown	49.0
SE	33	52.28
RSE (this work)	11	**54.34**
Universal Transformer	152	56.0
Human performance	-	86.0
GPT-3	175000	86.4

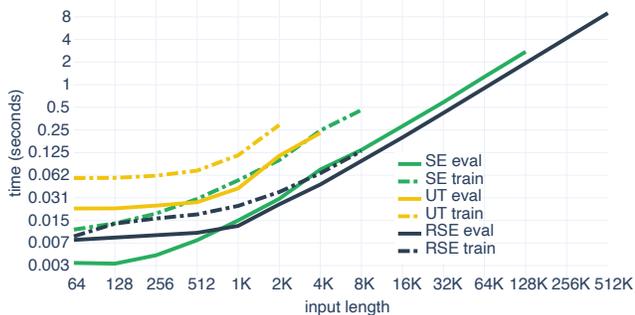Table 1: Accuracy on LAMBADA word prediction task.

Figure 5: Evaluation and training time on different input lengths (log-scale). Each model is evaluated on sequence lengths that fit in the 11GB of GPU memory. RSE has about 4x faster training and 2x faster inference than SE.
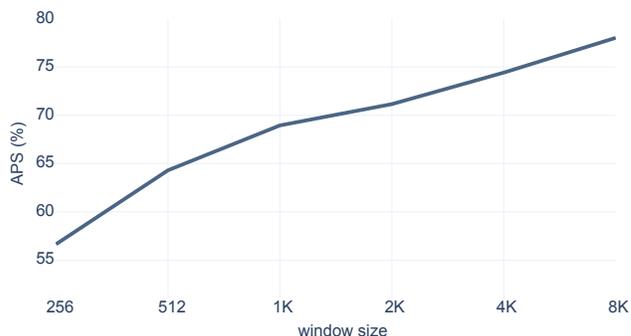

Figure 6: The MusicNet performance of our model on various window sizes. Using window sizes larger than 8K gives marginal accuracy improvement at a considerable increase in training time.

able parameters. SE and RSE networks have 384 feature maps and 2 Beneš blocks, with total parameter count 33M and 11M, respectively.

We evaluate sequence lengths that fit in the 11GB of GPU memory. The Residual Shuffle-Exchange network works faster and can be evaluated on 4x longer sequences than Shuffle-Exchange network and 128x longer sequences than the Universal Transformer.

MusicNet

The music transcription dataset MusicNet (Thickstun, Harchaoui, and Kakade 2017) consists of 330 classical music recordings paired with the MIDI transcriptions of their notes. The total length of the recordings is 34 hours, and it features 11 different instruments. The task is to classify what notes are being played at each time step given the waveform. As multiple notes can be played at the same time, this is a multi-label classification task.

For performing classification, regularly spaced windows of a given length are extracted from the waveform. We predict all the notes that are played at the midpoint of the extracted window.

We use an RSE model with two Beneš blocks with 192 feature maps. We experimentally found this to be the best configuration. To increase the training speed, we prepend two strided convolutions in front of that, see analysis of other options below. To obtain the note predictions, we use the element in the middle of the sequence output by the RSE model, linearly transform it to the 128 values, one for each note pitch, and apply the sigmoid cross-entropy loss function to perform multi-label classification.

For evaluating the model, we use the average precision score (APS), which is the area under the precision-recall curve. This metric is well suited to prediction tasks with imbalanced classes and is suggested for the MusicNet dataset in the original paper (Thickstun, Harchaoui, and Kakade 2017). We evaluate APS using the scikit-learn machine learning library (Pedregosa et al. 2011).

We train the model on different window sizes ranging from 128 to 8192 (see Fig. 6). We find that larger window

sizes invariably lead to better accuracy. The best APS score of 78.02% is obtained on length 8192.

The previous state-of-the-art was achieved by the Translation invariant net (Thickstun et al. 2018). They used task-specific handcrafted filterbanks to achieve invariance in representation with respect to pitch shifts of the input audio. It incorporates the prior knowledge of invariances in the problem domain to train on an augmented version of the MusicNet, where data is pitch-shifted by an integral number of semitones.

See Table 2 for a comparison with other works. A majority of the best results on the MusicNet have architectures that are specialized to use complex-valued data. This can give an advantage in sound processing tasks where data can be Fourier-transformed into a complex-valued representation before it is given as an input to the model. Trabelsi et al. (2018) achieves 72.90% APS with Deep Complex Network – a complex-valued convolution architecture. Its real-valued counterpart Deep Real Network achieves a lower score of 69.80% APS. cgRNN is an RNN that uses a recurrent cell with complex-valued transitions (Wolter and Yao 2018). It achieves 53% APS and uses only 2.36M parameters. Yang et al. (2020) proposed Complex Transformer that achieves 74.22% APS.

We have tested how prepending convolutions impact the speed and accuracy of the model. We find that one convolution gives the best results, although the differences are small. We chose to use two convolutions for a good balance be-

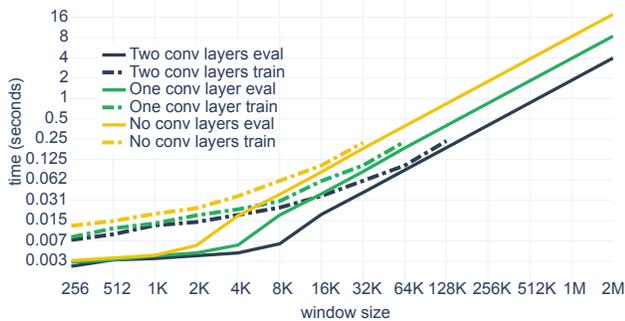| Model	Parameters (M)	APS (%)
cgRNN	2.36	53.00
Deep Real Network	10.0	69.80
Deep Complex Network	8.8	72.90
Complex Transformer	11.61	74.22
Translation-invariant net	unknown	77.3
RSE (this work)	3.06	**78.02**

Table 2: Performance on the MusicNet dataset.

Figure 7: Training and evaluation speed (log-scale) depending on the prepended convolution count on the MusicNet task.

Figure 8: Ablation experiments. The plot shows test error on the multiplication task length 128 vs training step.

tween speed and accuracy.

We use the batch size of one example in this test to see the sequence length limit our model can be trained and tested on a single GPU. Fig. 7 shows the training and evaluation speed of the model depending on the number of convolutions. Increasing the number of convolution layers improves training and evaluation speed, and the version with two convolutions can be trained on sequences up to length 128K. With 86 predictions per second as in the test set, our state-of-the-art model can perform inference 1.93 times faster than realtime. The training lines stop at the length at which the model does not fit in the GPU memory anymore. Testing lines reach a 2M technical limitation of our implementation.

In Appendix C, we provide accuracy measurements depending on the window size and the number of convolutions. We visualize the note predictions and describe the most common errors in Appendix D. Additional training details are provided in Appendix E.

Ablation Study

We have chosen the multiplication task as a showcase for the ablation study. It is a hard task which challenges every aspect of the model and performance differences are clearly observable. We use a model with 2 Beneš blocks, 192 feature maps and train it on length 128. We consider the following simplifications of the proposed architecture:

- removing LayerNorm (without LayerNorm)
- using ReLU instead of GELU
- removing the residual connection; the last equation of RSU becomes $[o_1, o_2] = c$ (without residual)
- setting the residual weight parameter $\sigma(S)$ to a constant 1 instead of a learnable parameter; the equation becomes $[o_1, o_2] = i + h \odot c$ (without residual scale)

We can see in Fig. 8 that the proposed baseline performs the best. Versions without residual connection or normalization do not work well. Residual weight parameter and GELU non-linearity give a smaller contribution to the model's performance.
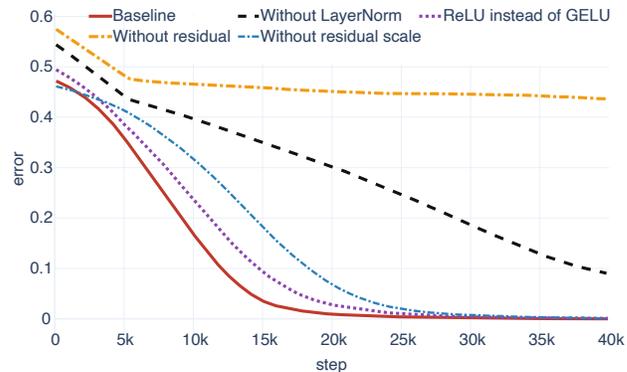
We investigate the effect of the RSU hidden layer size on performance. Parameter count and speed of the model is directly proportional to the hidden layer size; therefore, we want to select the smallest size, which gives a good performance. By default, we use $2m$ feature maps where m is the number of feature maps of the model. Versions with half as large or twice as large hidden layer size are explored. We discover that a larger hidden layer size leads to better performance. We consider the choice of $2m$ hidden layer size a good compromise between performance and parameter count. The experimental results for various hidden layer sizes are found in Appendix F.

We have performed ablation experiments also for LAMBADA and MusicNet tasks with similar conclusions, but the differences are much less pronounced.

Conclusions

We have proposed a simpler and faster version of the neural Shuffle-Exchange network. It has O($n \log n$) complexity and enables processing of sequences up to length 2 million where standard methods, like attention, fail. While keeping the overall successful connectivity structure of the Shuffle-Exchange network, we have shown that using residual connections instead of gated connections in its design, gives a significant boost to the training speed and achieved accuracy. Additionally, we have shown how to combine the model with strided convolutions that increase its speed and sequence length that can be processed.

The proposed model achieves state-of-the-art accuracy in recognizing musical notes directly from the waveform – a task where the ability to process long sequences is crucial. Notably, our architecture uses significantly fewer parameters than most of the previous best models for this task.

Our experiments confirm the Residual Shuffle-Exchange network as a useful building block for long sequence processing applications.

Acknowledgements

References

Al-Rfou, R.; Choe, D.; Constant, N.; Guo, M.; and Jones, L. 2018. Character-level language modeling with deeper self-attention. *arXiv preprint arXiv:1808.04444* .

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer Normalization. *arXiv preprint arXiv:1607.06450* .

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Beltagy, I.; Peters, M. E.; and Cohan, A. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150* .

Benetos, E.; Dixon, S.; Duan, Z.; and Ewert, S. 2019. Automatic Music Transcription: An Overview. *IEEE Signal Processing Magazine* 36: 20–30.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* .

Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734. Association for Computational Linguistics.

Chu, Z.; Wang, H.; Gimpel, K.; and McAllester, D. 2017. Broad Context Language Modeling as Reading Comprehension. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 52–57. ACL (Association for Computational Linguistics).

Dally, W. J.; and Towles, B. P. 2004. *Principles and practices of interconnection networks*. Elsevier.

Dehghani, M.; Gouws, S.; Vinyals, O.; Uszkoreit, J.; and Kaiser, Ł. 2018. Universal transformers. *arXiv preprint arXiv:1807.03819* .

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Freivalds, K.; and Liepins, R. 2018. Improving the neural GPU architecture for algorithm learning. *The ICML workshop Neural Abstract Machines & Program Induction v2 (NAMPI 2018)* .

Freivalds, K.; Ozoliņš, E.; and Šostaks, A. 2019. Neural Shuffle-Exchange Networks – Sequence Processing in O(n log n) Time. In *Advances in Neural Information Processing Systems 32*, 6626–6637. Curran Associates, Inc.

Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; and Dauphin, Y. N. 2017. Convolutional Sequence to Sequence Learning. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 1243–1252. PMLR.

Grefenstette, E.; Hermann, K. M.; Suleyman, M.; and Blunsom, P. 2015. Learning to Transduce with Unbounded Memory. In Cortes, C.; and Lee D.D. et al., eds., *Advances in Neural Information Processing Systems 28*, 1828–1836. Curran Associates, Inc.

Hendrycks, D.; and Gimpel, K. 2016. Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415* .

Huang, C.-Z. A.; Vaswani, A.; Uszkoreit, J.; Simon, I.; Hawthorne, C.; Shazeer, N.; Dai, A. M.; Hoffman, M. D.; Dinculescu, M.; and Eck, D. 2019. Music Transformer. In *International Conference on Learning Representations*.

Joulin, A.; and Mikolov, T. 2015. Inferring Algorithmic Patterns with Stack-Augmented Recurrent Nets. In Cortes, C.; and Lee D.D. et al., eds., *Advances in Neural Information Processing Systems 28*, 190–198. Curran Associates, Inc.

Kaiser, Ł.; and Bengio, S. 2016. Can Active Memory Replace Attention? In Lee, D.; and Luxburg U.V. et al., eds., *Advances in Neural Information Processing Systems 29*, 3781–3789. Curran Associates, Inc.

Kaiser, Ł.; and Sutskever, I. 2015. Neural GPUs learn algorithms. *arXiv preprint arXiv:1511.08228* .

Kalchbrenner, N.; Danihelka, I.; and Graves, A. 2015. Grid long short-term memory. *arXiv preprint arXiv:1507.01526* .

Kant, N. 2018. Recent Advances in Neural Program Synthesis. *arXiv preprint arXiv:1802.02353* .

Kitaev, N.; Kaiser, Ł.; and Levskaya, A. 2020. Reformer: The Efficient Transformer. In *International Conference on Learning Representations*.

Liu, L.; Jiang, H.; He, P.; Chen, W.; Liu, X.; Gao, J.; and Han, J. 2019. On the Variance of the Adaptive Learning Rate and Beyond. *arXiv preprint arXiv:1908.03265* .

Mikolov, T.; Grave, E.; Bojanowski, P.; Puhrsch, C.; and Joulin, A. 2018. Advances in Pre-Training Distributed Word Representations. In Calzolari, N.; and Choukri, Khalid et al., eds., *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA).

Paperno, D.; Kruszewski, G.; Lazaridou, A.; Pham, Q. N.; Bernardi, R.; Pezzelle, S.; Baroni, M.; Boleda, G.; and Fernández, R. 2016. The LAMBADA dataset: word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1525–1534. ACL (Association for Computational Linguistics).

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research* 12(Oct): 2825–2830.

Thickstun, J.; Harchaoui, Z.; Foster, D. P.; and Kakade, S. M. 2018. Invariances and Data Augmentation for Supervised Music Transcription. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Thickstun, J.; Harchaoui, Z.; and Kakade, S. M. 2017. Learning Features of Music from Scratch. In *International Conference on Learning Representations (ICLR)*.

Trabelsi, C.; Bilaniuk, O.; Zhang, Y.; Serdyuk, D.; Subramanian, S.; Santos, J. F.; Mehri, S.; Rostamzadeh, N.; Bengio, Y.; and Pal, C. J. 2018. Deep Complex Networks. *arXiv preprint arXiv:1705.09792* .

van den Oord, A.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A. W.; and Kavukcuoglu, K. 2016. WaveNet: A Generative Model for Raw Audio. In *SSW - the 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 2016*, 125.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is All you Need. In Guyon, I.; and Luxburg U.V. et al., eds., *Advances in Neural Information Processing Systems 30*, 5998–6008. Curran Associates, Inc.

Wang, S.; Li, B.; Khabsa, M.; Fang, H.; and Ma, H. 2020. Linformer: Self-Attention with Linear Complexity. *arXiv preprint arXiv:2006.04768* .

Wolter, M.; and Yao, A. 2018. Complex gated recurrent neural networks. In *Advances in Neural Information Processing Systems*, 10536–10546.

Xu, J.; Sun, X.; Zhang, Z.; Zhao, G.; and Lin, J. 2019. Understanding and Improving Layer Normalization. In *Advances in Neural Information Processing Systems*, 4383–4393.

Yang, M.; Ma, M. Q.; Li, D.; Tsai, Y. H.; and Salakhutdinov, R. 2020. Complex Transformer: A Framework for Modeling Complex-Valued Sequence. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4232–4236.

Yu, F.; and Koltun, V. 2015. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122* .

Zaheer, M.; Guruganesh, G.; Dubey, A.; Ainslie, J.; Alberti, C.; Ontanon, S.; Pham, P.; Ravula, A.; Wang, Q.; Yang, L.; et al. 2020. Big bird: Transformers for longer sequences. *arXiv preprint arXiv:2007.14062* .

Zaremba, W.; Mikolov, T.; Joulin, A.; and Fergus, R. 2016. Learning simple algorithms from examples. In *International Conference on Machine Learning*, 421–429.

Zaremba, W.; and Sutskever, I. 2015. Reinforcement Learning Neural Turing Machines-Revised. *arXiv preprint arXiv:1505.00521* .