

---

# Hierarchical Attention Encoder Decoder

---

**Asier Mujika**

Department of Computer Science  
ETH Zürich  
Zürich, Switzerland  
asierm@inf.ethz.ch

## Abstract

Recent advances in large language models have shown that autoregressive modeling can generate complex and novel sequences that have many real-world applications. However, these models must generate outputs autoregressively, which becomes time-consuming when dealing with long sequences. Hierarchical autoregressive approaches that compress data have been proposed as a solution, but these methods still generate outputs at the original data frequency, resulting in slow and memory-intensive models. In this paper, we propose a model based on the Hierarchical Recurrent Encoder Decoder (HRED) architecture. This model independently encodes input sub-sequences without global context, processes these sequences using a lower-frequency model, and decodes outputs at the original data frequency. By interpreting the encoder as an implicitly defined embedding matrix and using sampled softmax estimation, we develop a training algorithm that can train the entire model without a high-frequency decoder, which is the most memory and compute-intensive part of hierarchical approaches. In a final, brief phase, we train the decoder to generate data at the original granularity. Our algorithm significantly reduces memory requirements for training autoregressive models and it also improves the total training wall-clock time.

## 1 Introduction

Autoregressive modeling has been widely employed to model various types of sequential data, including language [1], images [2], and audio [3]. In recent years, autoregressive models have gained significant attention due to their impressive scaling properties [4, 5, 6]. By training on larger datasets, increasing model sizes, and utilizing prompt engineering<sup>1</sup>[7], these models can achieve outstanding results in tasks such as question answering[1], summarization[1], and image completion [8], among others.

Transformers [9] serve as the primary model for large-scale autoregressive learning. However, these models must operate at the same frequency as the original data, resulting in high computational costs. In language modeling, this problem is partially mitigated through tokenization[10], where frequent co-occurring character sequences are replaced by a single token. Nevertheless, models aiming to learn dependencies at the sentence level must still handle the original input data frequency. In domains with even higher-frequency inputs, such as images or sound, this issue becomes a primary bottleneck. A large enough model capable of capturing the data distribution would be too slow and memory-intensive to operate at the original data’s frequency. To put this in context, GPT-4, one of the most powerful language model trained to date, has a context window of 25,000 tokens, which would barely suffice to capture the individual pixels of a 92x92 RGB image in an autoregressive manner.

---

<sup>1</sup>Prompt engineering refers to modifying the context of a general autoregressive model to achieve a concrete goal.

To address this issue, we revisit the Hierarchical Recurrent Encoder-Decoder (HRED) architecture [11], which separates the computationally intensive high-frequency data from the more complex low-frequency interactions. This model divides the input sequence into non-overlapping sub-sequences that are encoded independently of each other. The resulting lower-frequency vector stream is then processed by a potentially larger model, no longer bottlenecked by the original data’s frequency. Finally, the output vectors are individually mapped back to the original data frequency. Although subsequent downsampling approaches have been proposed in the transformer literature, most of them feature an encoder or decoder that globally handles the entire original sequence, resulting in significant computational and memory cost. By encoding and decoding sub-sequences independently, the HRED model can circumvent these issues.

In this paper, we modify the HRED model to enhance its performance and propose a learning algorithm specifically designed for this architecture. Our main contributions include:

- Analyzing the different components of the HRED to identify which parts contribute the most to model performance.
- Based on these insights, we propose a modification of the HRED, termed the Hierarchical Attention Encoder-Decoder (HAED) architecture, that considerably improves the performance of the original model. Our model replaces the recurrent neural networks in the encoder and main model by MLPs and transformers, respectively.
- Introducing a learning algorithm for the HAED, capable of learning with low-frequency targets directly, which significantly reduces compute time and consequently allows for larger models and greater amounts of training data.

## 2 Related Work

Our work takes inspiration from the Hierarchical Recurrent Encoder-Decoder (HRED) model [11], which was initially proposed to independently encode user-generated queries and decode new queries from the context created by all previous queries. Our approach diverges by incorporating Transformers and MLPs instead of the original RNNs, applying the method to character and pixel modeling, and proposing a new learning algorithm specifically designed for this architecture.

Numerous hierarchical approaches have been suggested in the recurrent network literature, mainly to tackle the vanishing gradients problem [12, 13, 14, 15]. However, most of these approaches employ the same model for encoding and decoding, which prevents independent analysis of both mechanisms and the development of decoupled learning algorithms—two key contributions of our paper. Additionally, many of these studies attempt to learn the hierarchy itself [13, 15], a problem that we do not explore in this paper.

As the vanishing gradients problem is not as prominent in attention-based architectures as it is in RNNs, most attention-based hierarchical approaches concentrate on restricting the attention mask to attend only to sparse events in the sequence [16, 8, 17, 18]. While this reduces the computational costs of the attention mechanism, the model itself remains non-hierarchical and never reduces the input frequency. The Hourglass Transformer [19] is the most similar approach to ours. It features an encoder, decoder, and main model akin to our approach and the HRED model. However, since the encoder has global context, it cannot be interpreted as an embedding matrix, making our learning algorithm inapplicable to it. Furthermore, as both the encoder and decoder deal with global sequences, they experience typical issues with long sequence processing, such as large sequential computation and gradient decay with recurrent models or poor compute/memory scaling with very long sequences of attention-based models. Other similar approaches aimed at reducing sequence length for transformer architectures can be found in [20, 21, 22, 23].

A similar approach to our implicit embedding matrix learning algorithm was proposed by [24], which aims to maximize the information between the current global context and future encoded sub-sequences to learn unsupervised low-frequency representations of input data. Similarly, CLIP [25] applies this technique to match images to captions. They demonstrated that this approach outperforms learning an autoregressive decoder of captions from images, which is unsurprising considering our implicit embedding matrix interpretation. Their algorithm can be seen as simultaneously learning an autoregressive decoder of captions given images and images given captions, which mostly likely improves the learning signal.

The idea of replacing explicit embedding or output matrices with CNNs or RNNs was also investigated in [26] as a final step in language modeling to reduce the number of parameters since those matrices contained the majority of parameters in their experiments. Our work can be regarded as a generalization of this approach, where the output/embedding matrix is never actually materialized, allowing for training on open-vocabulary language modeling and image modeling.

Contemporary work to ours [27], has shown that a very similar model to the HRED and HAED, but with transformers for all models, can achieve state-of-the-art performance on many long-term dependency benchmarks. Based on our analysis of the different components, this is probably not the optimal way to distribute the compute. Still, our training algorithm can also be applied to that model.

### 3 Method

In this section we introduce the Hierarchical Attention Encoder-Decoder (HAED) architecture together with a learning algorithm specifically designed for this model, the Implicit Embedding Matrix (IEM) algorithm. The architecture is a modification of the Hierarchical Recurrent Encoder-Decoder (HRED) [11] that replaces the Recurrent Neural Networks (RNN) in the encoder and the main model by feed-forward MLPs and Transformers, respectively. The learning algorithm enables learning the model without a decoder, which is the most computationally expensive part of the model.

To disentangle hierarchy learning from hierarchy exploitation, we employ hard-coded hierarchies and focus solely on the second part in this paper. For language modeling, we aggregate characters at the word level, and for image modeling, we take a fixed number of pixels at each hierarchical level.

#### 3.1 Hierarchical Attention Encoder-Decoders

The HRED model divides the high-frequency input sequence into non-overlapping sub-sequences that are encoded individually and independently, see Equation 1. This gives rise to a new sequence that has a lower frequency than the original input. This new sequence is processed by the main model, to generate low-frequency predictions, see Equation 2. Finally, given the corresponding output from the main model, the decoder autoregressively generates the outputs at the frequency of the original sequence, see Equation 3. For simplicity, we use a fixed hierarchy of  $k$  steps, but translating these equations to input-dependent hierarchies would be straightforward. See Figure 1 for a diagram of the model.

$$\hat{x}_i = f_{enc}(x_{i \cdot k}, \dots, x_{k-1+i \cdot k}) \quad (1)$$

$$\hat{y}_i = f_{main}(\hat{x}_0, \dots, \hat{x}_i) \quad (2)$$

$$y_{i+j \cdot k} = f_{dec}(x_{j \cdot k}, \dots, x_{i+j \cdot k}, \hat{y}_j) \quad (3)$$

In the original HRED paper, the authors employ RNNs for all of these functions. Our Hierarchical Attention Encoder-Decoder (HAED) architecture introduces two changes to the HRED model. First, we replace the encoder model with a feed-forward MLP. As demonstrated in Section 4, the impact of  $f_{enc}$  on performance is minimal, and replacing it with an MLP significantly accelerates the encoding without any decrease in performance. Second, we utilize a more advanced Transformer architecture for  $f_{main}$ . This results in considerably better performance than the original RNN, without causing a major increase in compute or memory requirements, as this model operates at a lower frequency.

#### 3.2 Training on the embedding space

As shown in the Experiments section, the decoder, which operates at a high frequency, significantly affects the final performance. Consequently, most of the computation is required for the decoder. To address this, we introduce the Implicit Embedding Matrix (IEM) algorithm, that trains the model without the decoder, making predictions directly in the embedding space defined by the encoder.

We make a key observation for developing our learning algorithm: the encoder can be viewed as an implicit embedding matrix. As each sub-sequence is processed individually and without context, the encoder serves as a fixed mapping from a subsequence to a vector. This way, the encoder effectively defines an embedding matrix, where each entry corresponds to a possible subsequence, and the value of the entry is the output of the encoder.

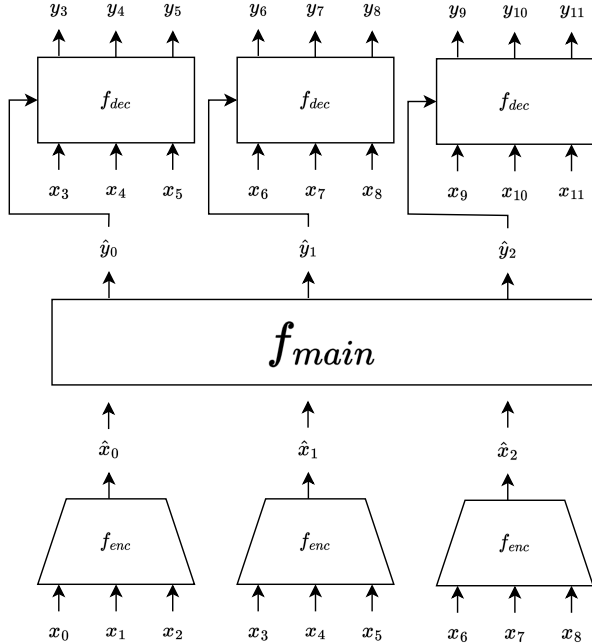


Figure 1: Diagram representing Equations 1, 2 and 3. In this example, the hierarchy aggregates the original inputs every 3 steps.  $f_{main}$  and  $f_{dec}$  are causally masked, to preserve the autoregressive information flow.

Another technique employed in our algorithm is commonly used in autoregressive modeling: using the transpose of the embedding matrix to compute individual token probabilities. Using the implicit embedding matrix interpretation, we express the probability assigned to the ground truth label as:

$$\text{logit}_i = f_{enc}(x_{i:k}, \dots, x_{k-1+i:k}) \cdot \hat{y}_i \quad (4)$$

$$p(x_{i:k}, \dots, x_{k-1+i:k} | x_0, \dots, x_{t-1}) = \frac{e^{\text{logit}_i}}{Z} \quad (5)$$

However, the softmax normalizing factor  $Z$  cannot be computed, as the embedding matrix is potentially exponential in size. For instance, in our image experiments, this matrix has  $2^{96}$  entries. Many approaches have been proposed for sampling and reweighting negative samples when the softmax matrix is too large to be dealt with explicitly. In this paper, we use the simplest approach of taking negative samples from the same batch to compute  $Z$ . While this is a biased estimate [28], it suffices to achieve good experimental results. We believe that better negative sampling techniques can considerably improve performance, but we leave that for future work.

As a final step to either sample or perform exact density estimation of the data, an actual decoder must be trained. However, as shown in Section 4, this can be done much faster after training with our algorithm.

## 4 Experiments

In this section, we will first experimentally evaluate the different components of the HRED and experimentally justify the introduction of the HAED architecture and next, we will assess the IEM algorithm and precisely measure the advantages it offers over end-to-end learning.

We will evaluate relatively small models (around 10 million parameters) on relatively large datasets, character-level WikiText-103 [29] and pixel-level <sup>2</sup> Imagenet-32 [30, 31]. We will do this in order

<sup>2</sup>Each pixel is represented by 3 tokens, each corresponding to a RGB channel

to avoid data reuse and thus, ignore overfitting, thereby reducing potential confounding effects. To further reduce possible confounding factors, we will use a hard-coded hierarchy. For character-level modeling we will encode/decode once per word and in the pixel-level modeling, we will encode/decode every 4 pixels, that is, every 12 inputs.

In order to improve the readability of this section but still ensure reproducibility, we move all the experimental details and hyperparameters to the Appendix.

#### 4.1 Experimental evaluation of HAED model

As a first step towards our HAED model we replace the LSTM in the main model of the HRED by a transformer. To validate this decision, we train two HRED models, one with a LSTM as a main model and one with a Transformer. Both models have roughly the same number of parameters. Table 1 summarizes our results. As expected, transformers performed much better than LSTMs. Unsurprisingly, the HRED with a Transformer considerably outperforms the original model.

Table 1: We train a LSTM based HRED and another one where the main model LSTM is replaced by a Transformer. Both models train on the same amount on data and have roughly the same number of parameters. The results are reported in bits per input token.

Dataset	HRED	HRED with Transformer
WikiText-103	1.52	1.37
Imagenet32	4.13	4.05

##### 4.1.1 Encoder Model

The encoder’s context is limited, which implies that its computation should be straightforward and not gain much from increased model capacity. To confirm this intuition, we train the attention based HRED from the previous section with LSTM encoders that have varying numbers of units, keeping the FLOPS/parameters of the rest of the model fixed, and plot the resulting impact on the final model performance. Figure 2 illustrates our findings, indicating that once the model has enough capacity to encode the sequence correctly, there is no further benefit in making the encoder larger.

Having this knowledge, we experiment with a simple 2-layer MLP as our encoder. This significantly speeds up the encoder, as it requires fewer FLOPS, allows for greater parallelization than sequential models and requires much less memory. Compared to a LSTM of equal final performance, it is faster and uses less memory. Therefore, for the remaining experiments, we will use this encoder architecture. We will refer to the model with a MLP encoder and Transformer main model as Hierarchical Attention Encoder Decoder (HAED).

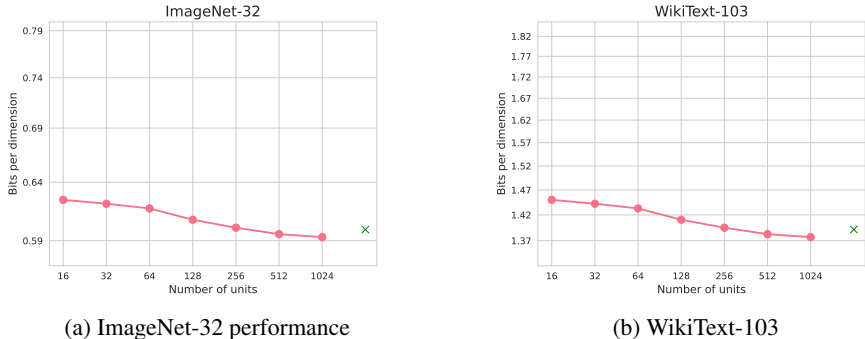


Figure 2: Model loss as a function of the number of units in the encoder. The "x" corresponds to the performance of the MLP model. The y-axes are set to the same range as in Figure 3.

### 4.1.2 Decoder Model

As with the encoder model, we want to measure the impact of increasing the decoder capacity on the overall model performance. Figure 3 shows the results of increasing the decoder capacity. In contrast to the encoder, we observe a continuous improvement on the loss as we increase the decoder size. However, this comes at a great computational cost as the decoder is ran at a much higher frequency than the main model. It can be seen that the decoder compute time quickly becomes the main bottleneck for training the model.

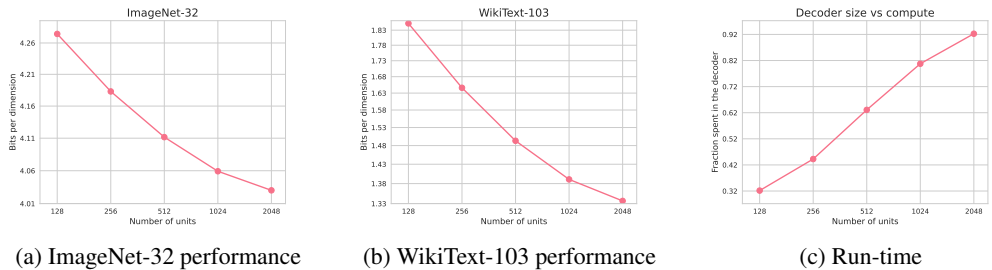


Figure 3: Model loss and percent of time spent in the decoder as a function of the number of units in it.

### 4.2 Decoupling

As illustrated in Figure 3, the majority of the training time is spent on training the decoder in larger models. Therefore, we aim to evaluate the benefits of pre-training with the IEM algorithm, allowing us to avoid using a decoder during most of the training process. In order to assess the advantages provided by our algorithm, we train an encoder and the main model using our auxiliary loss on half of the data from each dataset. Then, we fine-tune the model end-to-end with the decoder on the other half of the data. For comparison, we train a model end-to-end for the same amount of time as a baseline. Figure 4 demonstrates that our auxiliary loss improves the performance of the model, even when pre-training time is taken into account.

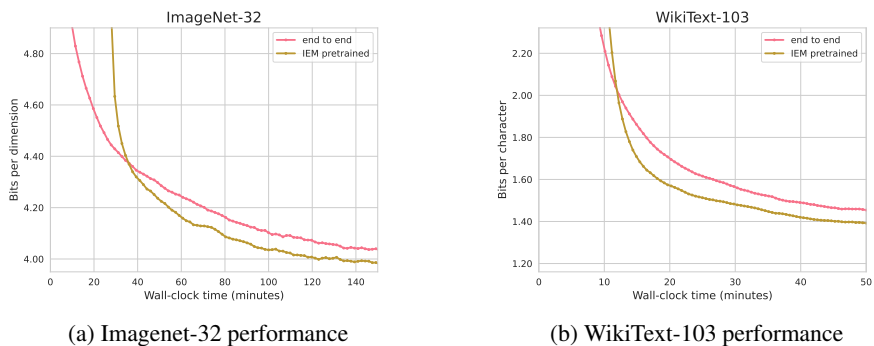


Figure 4: Model loss as a function of total wall-clock time, including pretraining.

## 5 Future Work

Our proposed approach offers many promising avenues for future work.

Firstly, we have only explored a single hierarchical level. While this has demonstrated significant advantages as shown in the sections above, adding more hierarchical levels could further improve the performance of the model.

Secondly, the way negative samples are selected is probably suboptimal. This selection method has been shown to have a considerable impact on the performance of models based on the sampled softmax

technique. In fact, the unigram distribution (the one we are sampling from in our experiments) tends to perform much worse than other context-dependent distributions. We believe that with carefully crafted negative samples, the pretraining loss will become even more aligned with the true loss and further improve performance.

Another avenue for future work could involve learning the hierarchies instead of hard-coding them. While there is a large body of knowledge addressing this question [13, 32, 21], we have not attempted to combine any of those techniques with our model for the sake of simplicity. However, we believe that exploring this direction presents a great opportunity for future work.

Lastly, a final direction of research that we find particularly exciting is sampling in the embedding space. Although learning in the embedding space with our algorithm greatly improves training, the model still needs to sample one data-point at a time in the high-frequency domain for inference. We believe our algorithm can be adapted to also sample in the low-frequency domain, which can then be decoded to the high-frequency domain in parallel. This would enable efficient autoregressive sampling in high-frequency domains like images or audio.

## 6 Limitations

While our approach and experiments have demonstrated promising results, there are several limitations worth noting.

Firstly, our study has been conducted using relatively small models, especially in comparison to the multi billion parameter models typically utilized to achieve state-of-the-art results. This size discrepancy raises questions about the potential scalability of our approach. In the context of very large models, it is conceivable that the decoder’s capacity may at some point cease to contribute to overall performance improvement. Under such circumstances, our Implicit Embedding Matrix (IEM) algorithm might lose its importance, as the main model would then become the primary computational bottleneck. However, it is in these exact situations that the Hierarchical Attention Encoder-Decoder (HAED) architecture could prove particularly valuable. This is due to its ability to operate at a lower frequency, thus reducing computational demands through architectural efficiency, as opposed to relying on the learning algorithm.

Another potential limitation of our work lies in the use of hard-coded hierarchies in our experiments. Although numerous techniques for learning hierarchies exist, none are flawless or demonstrate consistency on par with hard-coded hierarchies. It remains possible that our architecture might be particularly sensitive to errors within learned hierarchies. Therefore, the exploration of how our approach interacts with hierarchy learning methods is an interesting avenue for future research.

Despite these limitations, we believe that our proposed method opens up new horizons for improving the efficiency and effectiveness of large-scale autoregressive models.

## 7 Conclusion

We have presented a thorough experimental study of the different components of the Hierarchical Recurrent Encoder Decoder (HRED), which we have used to propose an improved model which we term the Hierarchical Attention Encoder Decoder (HAED).

Apart from that, we have also presented a learning algorithm that learns directly in the embedding space, rather than getting the learning signal from the high-frequency outputs, such as individual pixels. We show that by training with our algorithm, we can improve the performance and wall-clock training time of our model.

We believe that both our algorithm and model open many exciting possibilities for the use of large attention based models for autoregressive generation in domains where it wasn’t previously possible. Given the great scaling properties that these models have shown in language modeling, their applications for audio or image generation present exciting possibilities for the future.

## References

- [1] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.
- [2] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.
- [3] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.
- [4] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.
- [5] OpenAI. Gpt-4 technical report. March 2023.
- [6] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1):5232–5270, 2022.
- [7] Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*, 2023.
- [8] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International conference on machine learning*, pages 4055–4064. PMLR, 2018.
- [9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [10] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.
- [11] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *proceedings of the 24th ACM international on conference on information and knowledge management*, pages 553–562, 2015.
- [12] Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.
- [13] Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*, 2016.
- [14] Jan Koutnik, Klaus Greff, Faustino Gomez, and Juergen Schmidhuber. A clockwork rnn. In *International conference on machine learning*, pages 1863–1871. PMLR, 2014.
- [15] Bin Zhao, Xuelong Li, and Xiaoqiang Lu. Hsa-rnn: Hierarchical structure-adaptive rnn for video summarization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7405–7414, 2018.
- [16] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.
- [17] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.



- [18] Hongyu Ren, Hanjun Dai, Zihang Dai, Mengjiao Yang, Jure Leskovec, Dale Schuurmans, and Bo Dai. Combiner: Full attention transformer with sparse computation cost. *Advances in Neural Information Processing Systems*, 34:22470–22482, 2021.
- [19] Piotr Nawrot, Szymon Tworowski, Michał Tyrolski, Łukasz Kaiser, Yuhuai Wu, Christian Szegedy, and Henryk Michalewski. Hierarchical transformers are more efficient language models. *arXiv preprint arXiv:2110.13711*, 2021.
- [20] Zihang Dai, Guokun Lai, Yiming Yang, and Quoc Le. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. *Advances in neural information processing systems*, 33:4271–4282, 2020.
- [21] Piotr Nawrot, Jan Chorowski, Adrian Łańcucki, and Edoardo M Ponti. Efficient transformers with dynamic token pooling. *arXiv preprint arXiv:2211.09761*, 2022.
- [22] Jonathan H Clark, Dan Garrette, Iulia Turc, and John Wieting. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91, 2022.
- [23] Curtis Hawthorne, Andrew Jaegle, Cătălina Cangea, Sebastian Borgeaud, Charlie Nash, Mateusz Malinowski, Sander Dieleman, Oriol Vinyals, Matthew Botvinick, Ian Simon, et al. General-purpose, long-context autoregressive modeling with perceiver ar. In *International Conference on Machine Learning*, pages 8535–8558. PMLR, 2022.
- [24] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.
- [25] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
- [26] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.
- [27] Lili Yu, Dániel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. Megabyte: Predicting million-byte sequences with multiscale transformers. *arXiv preprint arXiv:2305.07185*, 2023.
- [28] Guy Blanc and Steffen Rendle. Adaptive sampled softmax with kernel based sampling. In *International Conference on Machine Learning*, pages 590–599. PMLR, 2018.
- [29] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.
- [30] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [31] Aäron van den Oord and Nal Kalchbrenner. Pixel rnn. 2016.
- [32] Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.
- [33] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [34] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- [35] Gábor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*, 2017.

## A Experimental details of the HAED model

For these experiments we use the Adam optimizer [33] with decoupled weight decay [34] and always clip the gradient norm to be at most 0.01. We use a learning rate of 0.002 for the encoder/decoder and 0.00035 for the main transformer model. We decay the learning rate to 0.05 of its initial value over training following a cosine schedule and use 2000 steps of linear warm-up for the transformer based experiments. We train all models for a single epoch. We use an embedding size of 10 for the high-frequency tokens. While this is very small, we didn't observe any advantage with larger embedding sizes. This could be because WikiText uses only characters and pixel-based tasks are actually encoding a real value (even though we treat it as a discrete value for autoregressive learning). For character-level modeling, we use words to split the original sequence. In order to avoid inefficiencies in our implementation when dealing with long words, we split words that are longer than 12 character long. For images, we split the image into subsequences of 12 tokens. We sample batches of 32 full images in ImageNet and of 256 words in WikiText.

In Section 4.1 we use the same encoder and decoder model, a 1024 unit LSTM. For the transformer, we use a small transformer, see Table 2 for the exact configuration. For the LSTM we use a 1500 unit LSTM, which has a similar number of parameters if we count only the recurrent ones. If we would take the LSTM input matrices into account, we would need to carefully tune input size to optimize performance. While only counting the recurrent parameters overestimates the performance of the LSTM, as it has more parameters than the transformer, the former still performs considerably better. We restrict the output of the LSTM to be of the same dimensionality as the transformer, such that the decoder still has the same number of parameters. We also found that the main LSTM could get unstable sometimes and we capped the input gate to be at most 1 minus the forget gate, which has been shown to have no major impact on performance, but offers more stability as this ensures the cell state stays between  $-1$  and  $1$  [35]. We used a learning rate of 0.002 for the main model LSTM which was found by grid search.

Table 2: The small Transformer architecture we use for most of our experiments.

Number of layers	6
Model dimensionality	356
Feed-forward dimensionality	1424
Number of heads	8
Dimensionality of heads	32
Positional Embeddings	Absolute learned embeddings
Positional Embedding size	100

In Section 4.1.1, we use the decoder and main transformer model as in the previous paragraph and only vary the number of units in the encoder. To always have the same input size to the main model, we pad the encoder output with 0's when the encoder output is smaller than the main models input and only take the first 356 units from the decoder when it is larger. For the MLP, we concatenate the 12 inputs into a vector and use the following 2 layer MLP:  $(12 \cdot 10) \rightarrow 256 \rightarrow ReLU \rightarrow 256 \rightarrow ReLU$ .

In Section 4.1.2, we use the same transformer as before, see Table 2. For the encoder we use the 2-layer MLP we described before. For the decoder, we use a standard LSTM with varying number of units.

## B Experimental details of the IEM algorithm

In this Section we use almost the same setup to the previous one. The only differences are that we use a transformer with 12 attention heads instead of 8, and a larger decoder with 2000 units. As negative samples we use the targets in the same batch and also sample 3 more batches that are only used for negative samples.