A Preliminaries on Autoregressive Transformers

Perplexity. Perplexity is a widely used metric for evaluating language models which encapsulates how well the model can predict a word. Formally, perplexity of a language model M is derived using the entropy formula as:

$$Perplexity(M) = 2^{H(L,M)} = 2^{-\sum_x L(x).log(M(x)))} \tag{1}$$

where L represents the ground-truth words. As seen, the perplexity is closely tied with the cross-entropy loss of the model, i.e., $H(L, M)$.

Parameter count. Contemporary autoregressive Transformer architectures can be divided into three main components, namely, the input embedding layer, hidden layers, and the final (softmax) projection layer. The embedding head often comprises look-up table based modules which map the input language tokens to vectors. These vectors then enter a stack of multiple hidden layers a.k.a, the decoder blocks. Each decoder block is made up of an attention layer and a feed-forward network. Once the features are extracted by the stack of decoder blocks, the final prediction is generated by passing through a final softmax projection layer. When counting the number of parameters in an autoregressive Transformer, the total parameters enclosed in the hidden layers is dubbed the decoder parameter count or equivalently, the non-embedding parameter count. These parameters are architecture dependent and do not change based on the underlying tokenization or the vocabulary size. The embedding parameter count, however, accounts for the parameters enclosed in the input embedding layer as well as the final softmax projection layer as they are both closely tied to the word embedding and vocabulary size. We visualize an autoregressive Transformer in Figure 15, where the orange blocks contain the decoder parameters and grey blocks hold the embedding parameters.

B Experimental Setup

Datasets. We conduct experiments on two datasets, WikiText-103 and LM1B. The datasets are tokenized using word-level and byte-pair encoding for models with Transformer-XL and GPT-2 backbones, respectively.

Training and Evaluation. We adopt the open-source code by [1] and [2] to implement the GPT-2 and Transformer-XL backbones, respectively. For each backbone and dataset, we use the same training setup for all models generated by NAS. Table 2 encloses the hyperparameters used for training all models in the experiments section of the paper. We follow the training hyperparameters, i.e., batch size, optimizer, learning rate values and scheduler, provided in NVIDIA's open-source repository [2]. Validation perplexity is measured over a sequence length of 192 and 32 tokens for WikiText-103 and LM1B datasets, respectively. Inference latency and peak memory utilization are measured on the target hardware for a sequence length of 192, averaged over 10 measurements. We utilize PyTorch's native benchmarking interface for measuring the latency and memory utilization of candidate architectures.

Table 2: LTS training hyperparameters for different backbones. Here, DO is the abbreviation used for dropout layers.

Backbone	Dataset	Tokenizer	# Vocab	Optim.	# Steps	Batch size	LR	Scheduler	Warmup	DO	Attn DO
Transformer-XL	WT103	Word	267735	LAMB [4]	4e4	256	1e-2	Cosine	1e3	0.1	0.0
	LM1B	Word	267735	Adam	1e5	224	2.5e-4	Cosine	2e4	0.0	0.0
GPT-2	WT103	BPE	50257	LAMB [4]	4e4	256	1e-2	Cosine	1e3	0.1	0.1
	LM1B	BPE	50257	LAMB [4]	1e5	224	2.5e-4	Cosine	2e4	0.1	0.1

Search Setup. Evolutionary search is performed for 30 iterations with a population size of 100; the parent population accounts for 20 samples out of the total 100; 40 mutated samples are generated per iteration from a mutation probability of 0.3; and 40 samples are created using crossover.

C How Good is the Decoder Parameters Proxy for Pareto-frontier Search?

Before we use the decoder parameter count as a proxy for perplexity in the inner loop of pareto-frontier search, we validate whether this proxy will actually help find pareto-frontiers which are close to the groundtruth. We first fully train all 1200 architectures sampled from the Transformer-XL backbone during evolutionary search (1). Using the validation perplexity obtained after full training, we rank all sampled architectures and extract the ground-truth pareto-frontier of perplexity versus latency. We train the models on the WikiText-103 dataset and benchmark Intel Xeon E5-2690 CPU as our target hardware platform for latency measurement in this experiment.

Figure 9 represents a scatter plot of the validation perplexity (after full training) versus latency for all sampled architectures during the search. The ground-truth pareto-frontier, by definition, is the lower convex hull of the dark navy dots, corresponding to models with the lowest validation perplexity for any given latency constraint. We mark the pareto-frontier points found by the training-free proxy with orange color. As shown, the architectures that were selected as the pareto-frontier by the proxy method are either on or very close to the ground-truth pareto-frontier.

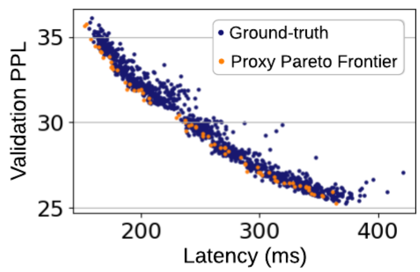


Figure 9: Perplexity versus latency pareto obtained from full training of 1200 architectures sampled during NAS on Transformer-XL backbone. Orange points are the pareto-frontier extracted using decoder parameter count proxy, which lies closely to the actual pareto-frontier. Decoder parameter count holds a SRC of 0.98 with the ground-truth perplexity after full training.

We define the mean average perplexity difference as a metric to evaluate the distance (d_{avg}) between the proxy and ground-truth pareto-frontier:

$$d_{avg} = \frac{1}{N} \sum_{i=1}^{N} \frac{|p_i - p_{gt,i}|}{p_{gt,i}} \tag{2}$$

Here, p_i denotes the i-th point on the proxy pareto front and $p_{gt,i}$ is the closest point, in terms of latency, to p_i on the ground-truth pareto front. The mean average perplexity difference for Figure 9 is $d_{avg} = 0.6\%$. This low difference further validates the effectiveness of our zero-cost proxy in correctly ranking the sampled architectures and estimating the true pareto-frontier. In addition to the low distance between the pareto-frontier estimated using decoder parameter count proxy and the ground-truth, our zero-cost proxy holds a high SRC of 0.98 over the entire pareto, i.e., all 1200 sampled architectures.

We further study the decoder parameter proxy in scenarios where the range of model sizes provided for search is limited. We categorize the total 1200 sampled architectures into different bins based on the decoder parameters. Figure 10 demonstrates the SRC between decoder parameter count proxy and the validation perplexity after full training for different model sizes. The proposed proxy provides a highly accurate ranking of candidate architectures even when exploring a small range of model sizes.

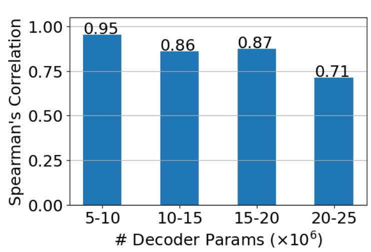
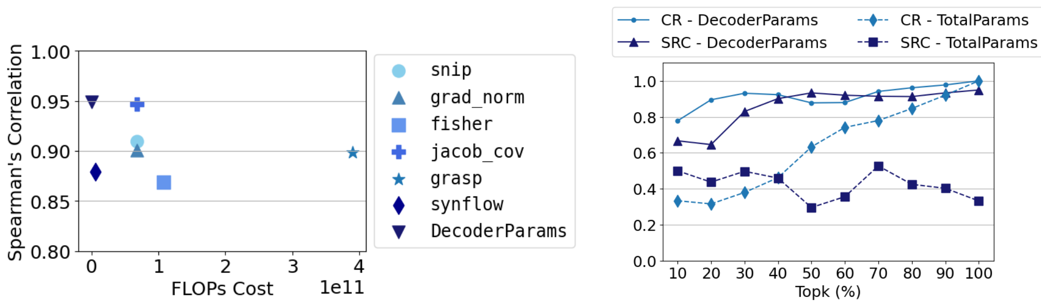


Figure 10: SRC between the decoder parameter count proxy and validation perplexity. Results are gathered on 1200 models grouped into four bins based on their decoder parameter count. Our proxy performs well even when exploring within a small range of model sizes.

D Analysis on Homogeneous Models

In this section, we evaluate the efficacy of the proposed proxies on the homogeneous search space, i.e., when all decoder layers have the same parameter configuration. In this scenario, the parameters are sampled from the valid ranges in Section 3 to construct one decoder block. This block is then replicated based on the selected $\mathrm{n}_{\mathrm{layer}}$ to create the Transformer architecture. In what follows, we provide experimental results gathered on 100 randomly sampled Transformer models from the Transformer-XL backbone with homogeneous decoder blocks, trained on WikiText-103.

► **Low-cost Proxies.** Figure 11a demonstrates the SRC between various low-cost methods and the validation perplexity after full training. On the horizontal axis, we report the total computation required for each proxy in terms of FLOPs. Commensurate with the findings on the heterogeneous models, we observe a strong correlation between the low-cost proxies and validation perplexity, with decoder parameter count outperforming other proxies. Note that we omit the `relu_log_det` method from Figure 11a as it provides a low SRC of 0.42 due to heavy reliance on ReLU activations.



(a) (b)

Figure 11: Experiments conducted on 100 randomly sampled Transformers with homogeneous decoder blocks, trained on WikiText-103. (a) SRC between ranking obtained from low-cost proxies and the ground-truth ranking after full training. The decoder parameter count obtains the best SRC with zero cost. (b) Performance of parameter count proxies. The decoder parameter count provides a very accurate ranking proxy with an SRC of 0.95 over all models.

► **Parameter Count.** As seen in Figure 11b, the total parameter count has a low SRC with the validation perplexity while the decoder parameter count provides an accurate proxy with an SRC of 0.95 over all architectures. These findings on the homogeneous search space are well-aligned with the observations in the heterogeneous space.

E How Does Model Topology Affect the Training-free Proxies?

Figure 13a shows the validation perplexity versus the aspect ratio of random architectures sampled from the Transformer-XL backbone and trained on WikiText-103. Here, the generated models span wide, shallow topologies (e.g., d_{model}=1024, n_{layer}=3) to narrow, deep topologies (e.g., d_{model}=128, n_{layer}=35). The maximum change in the validation perplexity for a given decoder parameter count is $< 7\%$ across wide range of aspect ratios $\sim 8 - 323$. Nevertheless, for the same decoder parameter count budget, the latency can vary by $1.3\times$ and the peak memory utilization by $2.0\times$ as shown in Figure 13b,13c, respectively.

For deeper architectures (more than 40 layers) with the Transformer-XL backbone, we observe an increase in the validation perplexity, which results in a deviation from the pattern in Figure 13a. This observation is associated with the inherent difficulty in training deeper architectures, which can be mitigated with proposed techniques in the literature [3]. Nevertheless, such deep models have a high latency, which makes them unsuitable for lightweight inference. For the purposes of hardware-aware and efficient Transformer NAS, our search-space contains architectures with less than 16 layers. In this scenario, the decoder parameter count proxy holds a very high correlation with validation perplexity, regardless of the architecture topology as shown in Figure 13a.

F 3D Pareto Visualization

Figure 14 visualizes the 3-dimensional pareto for the GPT-2 backbones. Here, the black and blue points denote the regular and pareto-frontier architectures, respectively. The pair of red dots are architectures which match in both memory and decoder parameter count ($\sim$ perplexity). However, as shown, their latency differs by $2\times$. The pair of green points correspond to models with the same decoder parameter count ($\sim$ perplexity) and latency, while the memory still differs by 30MB, which is non-negligible for memory-constrained application. In a 2-objective pareto-frontier search of perplexity versus memory (or latency), each pair of red (or green) dots will result in similar evaluations. While in reality, they have very different characteristics in terms of the overlooked metric. This experiment validates the need for multi-objective pareto-frontier search, which simultaneously takes into account multiple hardware performance metrics.

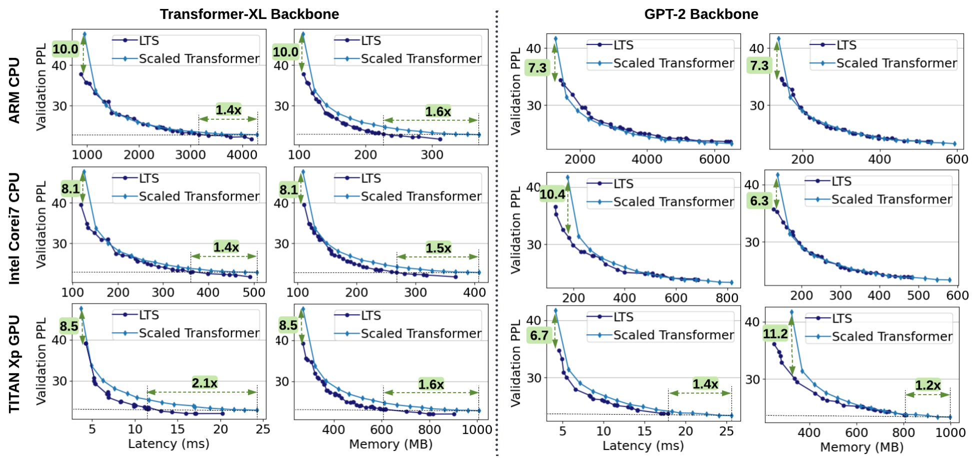
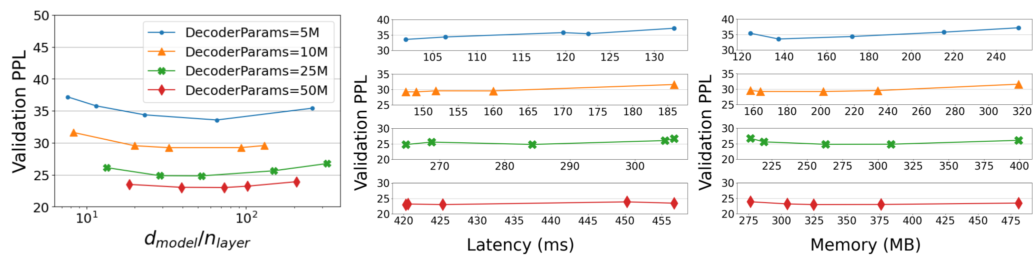


Figure 12: 2D visualization of the perplexity versus latency and memory pareto-frontier found by LTS and scaled backbone models with varying number of layers. All models are trained on the WikiText-103 dataset. The architectural parameters for all models are enclosed in Appendix H.



(a) (b) (c)

Figure 13: Validation perplexity after full training versus (a) the width-to-depth aspect ratio, (b) latency, and (c) peak memory utilization. Models are randomly generated from the Transformer-XL backbone and trained on WikiText-103. For a given decoder parameter count, we observe low variation in perplexity across different models, regardless of their topology. The topology, however, significantly affects the latency and peak memory utilization for models with the same perplexity.

G LTS Performance Comparison on WikiText-103

We compare the pareto-frontier architectures found by LTS with the baseline after full training on the WikiText-103 dataset in Figure 12. Commensurate with the findings on the LM1B dataset, the NAS-generated models outperform the baselines in at least one of the three metrics, i.e., perplexity, latency, and peak memory utilization. We note that the gap between the baseline models and those obtained from NAS is larger when training on the LM1B dataset. This is due to the challenging nature of LM1B, which exceeds the WikiText-103 dataset size by $\sim 10\times$. Thus, it is harder for hand-crafted baseline models to compete with the optimized LTS architectures on LM1B.

On the Transformer-XL backbone, the models on LTS pareto-frontier for the ARM CPU have, on average, 3.8% faster runtime and 20.7% less memory under the same validation perplexity budget. On the Corei7, the runtime and memory savings increase to 13.2% and 19.6%, respectively, while matching the baseline perplexity. We achieve our highest benefits on TITAN Xp GPU where the pareto-frontier of LTS has on average 31.8% lower latency and 21.5% less memory. Notably, the validation perplexity of the baseline 16-layer Transformer-XL base can be achieved with a lightweight model with $2.1\times$ less latency while consuming $1.6\times$ less memory at runtime.

On the GPT-2 backbone, LTS achieves $6.3 - 11.2$ lower perplexity in the low-latency-and-memory regime. As we transition to larger models and higher latency, our results show that the GPT-2 architecture is nearly optimal on WikiText-103 when performing inference on a CPU. The benefits are more significant when targeting a GPU; For any given perplexity achieved by the baseline, LTS pareto-frontier on TITAN Xp delivers, on average, 9.0% lower latency and 4.5% lower memory. Therefore, the perplexity and memory of the baseline 16-layer GPT-2 can be achieved by a new model that runs $1.4\times$ faster and consumes $1.2\times$ less memory on TITAN Xp.

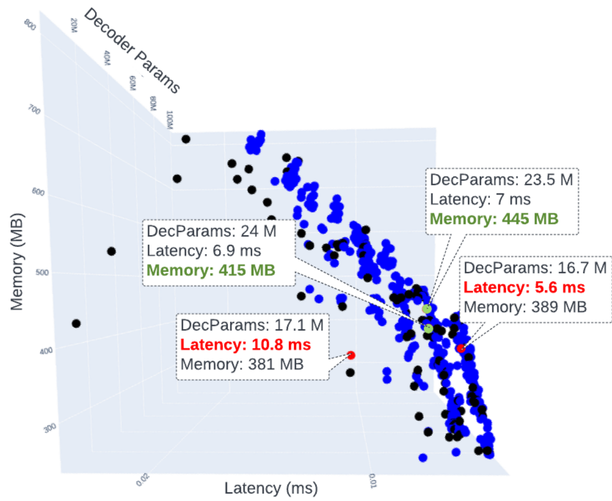


Figure 14: 3D visualization of our multi-objective NAS for the GPT-2 backbone on TITAN Xp GPU. Architectures with similar memory and decoder parameter count can result in drastically different runtimes (up to $2\times$ difference). Similarly, architectures with similar decoder parameter count and latency may have different peak memory utilization. Therefore, it is important to perform multi-objective NAS where several hardware characteristics are simultaneously taken into account when extracting the pareto-frontier.

H Architecture Details

Tables 3, 4, 5, 6 enclose the architecture parameters for the baseline and NAS-generated models in Figure 8 for Transformer-XL and GPT-2 backbones. For each target hardware, the rows of the table are ordered based on increasing decoder parameter count (decreasing validation perplexity). For all models, $d_{head}=d_{model}/n_{head}$, the adaptive input embedding factor is set to $k = 4$, and $d_{embed}=d_{model}$.

I Ethics Statement and Broader Impact

We provide an extremely lightweight method for NAS on autoregressive Transformers. Our work is likely to increase the adoption of NAS in the NLP domain, providing several prevalent benefits:

Firstly, a more widespread adoption of automated techniques, e.g., NAS eliminates the need for laborious trials and error for manual design of Transformer architectures, freeing up hundreds of hours of man-power as well as computational resources. Secondly, automating architecture design can trigger generation of new models with superior performance, which benefits the ever-growing applications of NLP in the everyday life. Finally, by making the search algorithm efficient, we ensure it can be accessible to the general scientific public without need for any expensive mode training, thereby minimizing the unwanted byproducts of the Deep Learning era such as the carbon footprint, and power consumption.

While the benefits of automation in NLP are plenty, it can lead to potential side-effects that have not been yet fully unveiled. Since our work advances the use of NAS in the NLP design pipeline, there is need for careful scrutiny of the models which have been automatically designed with respect to aspects such as bias, misinformation, and nefarious activity, to name a few.

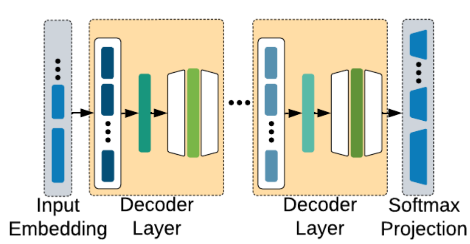


Figure 15: High-level visualization of different components in autoregressive Transformers. Here, the parameters enclosed in the orange blocks are counted as decoder parameters, while the parameters contained in the gray boxes denote the embedding parameter count.

References

[1] Hugging Face. Openai gpt2 by hugging face. `https://huggingface.co/docs/transformers/model_doc/gpt2`.

[2] NVIDIA. Transformer-xl for pytorch. `https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/LanguageModeling/Transformer-XL`.

[3] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers. *arXiv preprint arXiv:2203.00555*, 2022.

[4] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019.

Table 3: Detailed architectural parameters for all models in Figure 8 with Transformer-XL backbone.

		n_{layer}	d_{model}	n_{head}	d_{inner}	DecoderParams (M)
baseline		∈[1,16]	512	8	2048	-
ARM	M1	2	512	[2, 2]	[1216, 1280]	3.2
	M2	3	320	[2, 4, 2]	[1472, 2368, 3392]	5.5
	M3	2	512	[2, 2]	[2560, 2176]	5.5
	M4	2	512	[2, 2]	[3904, 1792]	6.5
	M5	2	640	[2, 2]	[3520, 3456]	9.8
	M6	2	832	[2, 2]	[3264, 3968]	13.1
	M7	2	704	[8, 2]	[3904, 3968]	13.4
	M8	2	960	[2, 2]	[3648, 3968]	15.9
	M9	2	960	[2, 2]	[3904, 3968]	16.4
	M10	3	960	[2, 2, 2]	[1856, 2368, 3392]	16.5
	M11	3	960	[2, 4, 2]	[3328, 2368, 3200]	19.6
	M12	3	832	[2, 2, 2]	[3904, 3968, 3008]	19.7
	M13	3	960	[2, 2, 2]	[3904, 3584, 3456]	22.9
	M14	3	960	[4, 2, 2]	[3648, 3584, 3584]	23.3
	M15	3	960	[2, 2, 8]	[4032, 3968, 3904]	26.6
	M16	4	896	[4, 2, 8, 2]	[3904, 3008, 3520, 3584]	29.7
	M17	4	960	[2, 2, 2, 2]	[3840, 3904, 3520, 3072]	30.0
	M18	4	960	[2, 2, 2, 2]	[4032, 3648, 3136, 4032]	31.0
	M19	4	960	[2, 2, 4, 2]	[3904, 3968, 3840, 3584]	32.5
	M20	4	960	[8, 8, 8, 4]	[4032, 3968, 2880, 3200]	35.7
	M21	4	960	[8, 2, 4, 8]	[4032, 3584, 3840, 3584]	35.7
	M22	5	960	[2, 2, 2, 2, 2]	[3904, 3968, 3264, 3456, 3200]	37.3
	M23	5	960	[2, 2, 2, 8, 2]	[3904, 3648, 3136, 3648, 3840]	39.9
	M24	6	960	[2, 2, 2, 2, 2, 8]	[3328, 2624, 3392, 2944, 3008, 3904]	42.5
	M25	6	960	[2, 4, 2, 2, 2, 2]	[2112, 3840, 3328, 3264, 3968, 3648]	43.1
	M26	6	960	[2, 2, 2, 2, 2, 4]	[3968, 3968, 3456, 3456, 3776, 2432]	44.8
	M27	6	960	[2, 2, 4, 2, 8, 8]	[3584, 2624, 3392, 3968, 3008, 3328]	46.3
	M28	6	960	[2, 4, 2, 2, 8, 2]	[3904, 3008, 3392, 3648, 3392, 3584]	46.4
	M29	6	960	[8, 8, 2, 4, 2, 4]	[3904, 3648, 3136, 3648, 3200, 3840]	49.7
	M30	6	960	[2, 4, 8, 4, 2, 8]	[3904, 3008, 3392, 3200, 3968, 3904]	49.7
	M31	6	960	[8, 4, 8, 4, 2, 8]	[3904, 3648, 3392, 3200, 3968, 3840]	52.7
	M32	8	896	[4, 2, 2, 4, 4, 2, 4, 8]	[3584, 3968, 3392, 3904, 2240, 1856, 2560, 3264]	53.1
	M33	8	896	[4, 2, 2, 2, 4, 4, 4, 2]	[3584, 3584, 3520, 2368, 2752, 4032, 3520, 3264]	54.7
	M34	8	960	[2, 4, 4, 4, 4, 8, 2, 2]	[3968, 3584, 3520, 3072, 3968, 4032, 1856, 3712]	62.5
	M35	9	896	[4, 2, 4, 4, 8, 2, 8, 8, 2]	[3840, 3136, 3520, 2880, 3200, 3008, 3328, 2560, 3136]	63.4
	M36	9	960	[4, 4, 8, 2, 2, 2, 8, 8, 2]	[2112, 3008, 3520, 3648, 3968, 4032, 1984, 3200, 3520]	68.0
	M37	9	960	[8, 2, 4, 2, 8, 8, 8, 2, 2]	[3968, 3008, 3520, 3200, 3200, 4032, 1984, 2816, 3520]	69.8
	M38	12	832	[2, 4, 4, 2, 2, 8, 8, 8, 4, 4, 2, 8]	[3136, 2112, 2112, 2368, 2752, 2432, 2432, 2176, 3456, 3712, 2880, 3712]	70.4
	M39	12	832	[4, 4, 4, 2, 2, 8, 4, 8, 2, 8, 2, 8]	[3136, 3968, 2112, 2368, 3072, 2240, 2624, 2112, 3456, 3072, 2880, 3264]	72.1
Corei7	M1	2	384	[2, 2]	[896, 2816]	3.4
	M2	2	576	[2, 2]	[1792, 2816]	6.1
	M3	2	832	[2, 2]	[1728, 1536]	6.5
	M4	2	576	[2, 2]	[1408, 3776]	6.7
	M5	2	768	[2, 2]	[2112, 3584]	9.7
	M6	2	768	[2, 2]	[3776, 1920]	9.7
	M7	2	832	[2, 2]	[3776, 3392]	13.0
	M8	2	960	[2, 4]	[1984, 3840]	13.0
	M9	2	832	[2, 2]	[3968, 3584]	13.7
	M10	2	960	[2, 2]	[3904, 3904]	16.2
	M11	2	960	[8, 8]	[3968, 3584]	19.4
	M12	3	960	[2, 2, 4]	[2176, 3840, 2880]	19.6
	M13	3	896	[2, 2, 2]	[2304, 3904, 3904]	19.9
	M14	3	960	[2, 2, 4]	[3776, 2880, 3904]	22.8
	M15	3	960	[2, 8, 2]	[3840, 3840, 3904]	26.0
	M16	3	960	[2, 2, 8]	[3968, 3904, 3904]	26.3
	M17	3	960	[2, 8, 8]	[3904, 3840, 3904]	27.9
	M18	4	960	[2, 4, 2, 2]	[3904, 2112, 4032, 3584]	29.3
	M19	4	960	[2, 2, 2, 4]	[2112, 3840, 3904, 3904]	29.5
	M20	4	960	[2, 2, 2, 4]	[3904, 3776, 3904, 3904]	32.9
	M21	4	960	[2, 4, 8, 4]	[3776, 3392, 3520, 3904]	33.6
	M22	5	960	[2, 2, 2, 2, 2]	[3776, 1984, 3904, 3904, 3456]	35.8
	M23	5	960	[2, 4, 2, 4, 2]	[3968, 3584, 3520, 3904, 3200]	39.3
	M24	5	960	[2, 4, 4, 4, 2]	[3776, 3840, 3904, 3904, 3968]	42.2
	M25	6	960	[2, 4, 2, 4, 2, 4]	[3776, 2112, 4032, 3584, 3200, 4032]	45.4
	M26	6	960	[2, 4, 4, 2, 2, 4]	[3776, 3840, 3904, 3904, 3008, 2304]	45.4
	M27	6	960	[2, 4, 2, 4, 4, 4]	[3776, 3840, 3904, 4032, 3648, 2432]	47.7
	M28	6	960	[4, 2, 8, 4, 2, 2]	[3840, 3712, 3520, 4032, 3200, 4032]	49.7
	M29	8	960	[2, 2, 2, 4, 2, 4, 8, 2]	[3392, 1792, 3904, 3904, 3200, 2432, 1792, 2496]	52.1
	M30	7	960	[2, 2, 8, 4, 2, 2, 4]	[3776, 3840, 3904, 1856, 3072, 3648, 4032]	53.8
	M31	8	960	[2, 2, 4, 4, 2, 4, 8, 2]	[3776, 3008, 4032, 3904, 3520, 3136, 1984, 3648]	60.5
	M32	8	960	[8, 2, 2, 4, 8, 4, 4, 8]	[3776, 3008, 3904, 3904, 2176, 4032, 4032, 3648]	67.1
	M33	9	960	[4, 2, 4, 4, 4, 4, 8, 8, 2]	[3840, 3136, 3520, 4032, 3200, 4032, 3648, 2112, 2368]	69.8
	M34	9	960	[8, 2, 8, 8, 2, 4, 8, 2, 2]	[3520, 3008, 2880, 4032, 3200, 2432, 4032, 3904, 3136]	71.5
	M35	13	768	[2, 8, 2, 4, 2, 2, 4, 2, 2, 8, 8, 8, 4]	[3776, 2112, 1600, 3904, 3840, 2880, 2304, 3200, 2048, 2944, 2816, 3328, 3968]	73.3

Table 4: Detailed architectural parameters for all models in Figure 8 with Transformer-XL backbone.

		n_{layer}	d_{model}	n_{head}	d_{inner}	DecoderParams (M)
	baseline	$\in$[1,16]	8	2048	512	-
TITAN Xp	M1	2	384	[2, 2]	[1152, 2432]	3.3
	M2	2	576	[2, 2]	[2048, 1728]	5.1
	M3	2	512	[2, 2]	[2368, 3072]	6.2
	M4	2	448	[8, 2]	[2944, 3008]	6.8
	M5	2	832	[8, 2]	[3264, 3072]	13.2
	M6	2	768	[2, 2]	[3968, 4032]	13.3
	M7	2	896	[8, 4]	[4032, 2880]	15.8
	M8	2	960	[2, 2]	[3840, 3968]	16.2
	M9	2	960	[4, 8]	[3968, 3008]	17.1
	M10	2	960	[4, 8]	[3968, 3648]	18.3
	M11	3	960	[2, 2, 2]	[3584, 3072, 2624]	19.7
	M12	3	896	[2, 2, 2]	[3840, 2880, 3840]	20.7
	M13	3	896	[8, 4, 8]	[4032, 2112, 3392]	22.9
	M14	3	960	[4, 2, 2]	[3840, 3008, 3840]	23.0
	M15	3	960	[2, 2, 8]	[3584, 4032, 4032]	26.1
	M16	3	960	[2, 2, 8]	[4032, 4032, 3840]	26.6
	M17	3	960	[8, 2, 8]	[4032, 4032, 3520]	27.8
	M18	3	960	[8, 4, 8]	[4032, 4032, 4032]	29.4
	M19	4	896	[4, 4, 8, 8]	[4032, 3456, 3328, 3392]	32.4
	M20	4	960	[4, 2, 8, 8]	[3840, 3008, 3328, 3584]	33.2
	M21	4	960	[4, 2, 4, 4]	[3840, 4032, 3904, 4032]	34.7
	M22	4	960	[2, 2, 8, 8]	[4032, 3968, 3904, 3840]	36.4
	M23	5	960	[4, 2, 4, 4, 8]	[3840, 3008, 3392, 2496, 4032]	39.0
	M24	5	960	[2, 2, 4, 4, 4]	[3968, 4032, 3328, 4032, 2752]	39.7
	M25	5	960	[2, 4, 2, 2, 8]	[3968, 3968, 3840, 4032, 3904]	43.4
	M26	5	960	[4, 2, 8, 8, 8]	[3840, 3008, 3840, 3328, 3968]	43.8
	M27	5	960	[8, 2, 8, 8, 4]	[4032, 3008, 3840, 3904, 3968]	45.3
	M28	6	896	[2, 2, 4, 4, 2, 2]	[3840, 3968, 3840, 3328, 3904, 3904]	45.5
	M29	6	896	[8, 4, 8, 4, 8, 8]	[3328, 2112, 3392, 3904, 3328, 3264]	46.2
	M30	6	960	[4, 2, 2, 4, 2, 8]	[3840, 3008, 3840, 3904, 4032, 3392]	49.1
	M31	6	960	[4, 8, 8, 4, 8, 4]	[3072, 3584, 3392, 3840, 3328, 3712]	51.3
	M32	6	960	[2, 4, 8, 8, 4, 2]	[3840, 3968, 3840, 3328, 4032, 3776]	52.4
	M33	6	960	[4, 8, 8, 8, 4, 4]	[3840, 3584, 3392, 3328, 3968, 3776]	53.1
	M34	6	960	[4, 8, 8, 8, 8, 2]	[3840, 3840, 3392, 3840, 3328, 3712]	53.9
	M35	7	960	[4, 8, 8, 8, 8, 2, 8]	[3840, 3968, 3840, 3328, 3968, 3328, 4032]	64.7
	M36	8	960	[4, 2, 8, 8, 8, 4, 8, 8]	[3840, 3968, 3840, 3328, 3072, 3328, 4032, 3072]	70.1
	M37	10	896	[8, 8, 8, 2, 8, 2, 2, 2, 8, 2]	[3840, 3072, 3840, 2560, 3648, 3328, 3840, 3008, 2880, 3328]	74.2
	M38	9	960	[8, 8, 8, 4, 4, 8, 8, 4, 2]	[2752, 3456, 2880, 3904, 2752, 3904, 4032, 3264, 3136]	74.4
	M39	10	896	[8, 4, 8, 8, 8, 2, 8, 2, 4, 8]	[4032, 3008, 3840, 2560, 3904, 3904, 3072, 3264, 2368, 2496]	75.4
	M40	12	832	[2, 4, 8, 8, 8, 8, 8, 8, 8, 8, 4, 2]	[3840, 2816, 2112, 3584, 3648, 2432, 2304, 3008, 2880, 1664, 2432, 3776]	77.7

Table 5: Detailed architectural parameters for all models in Figure 8 with GPT-2 backbone.

		n_{layer}	d_{model}	n_{head}	d_{inner}	DecoderParams (M)
	baseline	$\in$[1,16]	8	2048	512	-
TITAN Xp	M1	3	256	[2, 2, 2]	[3072, 3776, 3904]	6.3
	M2	2	448	[2, 2]	[3456, 3776]	8.1
	M3	2	448	[2, 4]	[4032, 3904]	8.7
	M4	3	384	[2, 2, 2]	[3072, 2176, 4032]	8.9
	M5	2	576	[2, 2]	[3456, 3584]	10.8
	M6	4	448	[2, 2, 2, 2]	[4032, 3904, 1920, 3072]	14.8
	M7	4	512	[2, 2, 4, 2]	[3904, 3136, 1280, 2624]	15.4
	M8	2	832	[8, 2]	[3456, 3584]	17.3
	M9	2	960	[2, 8]	[3456, 3648]	21.0
	M10	2	960	[2, 2]	[3968, 3584]	21.9
	M11	5	640	[2, 2, 2, 2, 2]	[4032, 2560, 2176, 2304, 3136]	26.4
	M12	3	832	[2, 8, 4]	[3840, 3840, 3776]	27.4
	M13	5	704	[2, 2, 2, 4, 4]	[2368, 3648, 1856, 3712, 3200]	30.8
	M14	3	960	[2, 2, 2]	[3584, 3648, 4032]	32.7
	M15	3	960	[2, 2, 2]	[3904, 3520, 4032]	33.1
	M16	6	640	[2, 2, 2, 2, 2, 2]	[2624, 2560, 2880, 3776, 3648, 3840]	34.6
	M17	4	896	[2, 2, 4, 2]	[4032, 3712, 3328, 3072]	38.2
	M18	5	832	[2, 2, 2, 4, 4]	[3392, 3648, 2880, 3712, 3200]	41.9
	M19	4	960	[2, 2, 4, 2]	[3904, 3136, 3328, 3776]	42.0
	M20	4	960	[8, 8, 2, 4]	[3904, 3712, 4032, 3776]	44.4
	M21	6	832	[2, 2, 4, 2, 2, 2]	[3904, 3456, 4032, 1792, 3072, 2496]	47.9
	M22	5	896	[4, 2, 2, 2, 4]	[3968, 3200, 3840, 3328, 3648]	48.3
	M23	5	960	[2, 2, 2, 2, 2]	[3904, 3264, 3328, 3776, 3392]	52.4
	M24	5	960	[2, 2, 4, 2, 2]	[3584, 3456, 3776, 2944, 4032]	52.7
	M25	5	960	[2, 8, 2, 4, 2]	[3904, 3648, 4032, 3776, 3968]	55.6
	M26	6	960	[8, 8, 2, 2, 2, 2]	[3904, 2560, 2880, 3776, 2240, 3840]	59.1
	M27	6	960	[2, 2, 2, 4, 2, 2]	[2496, 3456, 3328, 3904, 3968, 2944]	60.8
	M28	6	960	[4, 2, 4, 4, 2, 8]	[4032, 3456, 3328, 3776, 4032, 2752]	63.2
	M29	6	960	[2, 2, 2, 4, 4, 4]	[3968, 3648, 3840, 3776, 3584, 2624]	63.4
	M30	7	960	[2, 2, 2, 4, 2, 4, 2]	[3904, 2368, 4032, 3008, 3520, 2944, 2496]	68.7
	M31	7	960	[2, 2, 4, 2, 2, 2, 4]	[3072, 3648, 3520, 3584, 3136, 1984, 3584]	69.1
	M32	7	960	[4, 2, 2, 2, 8, 2, 2]	[3712, 3648, 3584, 3520, 2752, 3008, 3392]	71.2
	M33	8	960	[2, 4, 4, 2, 2, 2, 2, 2]	[3904, 2816, 3072, 1920, 3328, 3456, 2304, 2368]	74.1
	M34	8	960	[2, 2, 2, 4, 2, 2, 8, 2]	[3520, 2368, 4032, 1792, 3200, 3776, 3200, 3648]	78.6
	M35	8	960	[4, 2, 4, 4, 8, 8, 4, 2]	[3520, 3712, 3328, 3776, 3200, 2752, 3200, 2112]	78.7
	M36	8	960	[8, 4, 2, 8, 2, 2, 2, 2]	[3520, 3840, 3328, 3776, 3200, 3776, 3968, 3648]	85.4
	M37	10	960	[2, 8, 2, 4, 2, 2, 4, 2, 8, 8]	[3648, 2560, 3776, 1792, 3968, 2752, 3200, 2368, 4032, 2368]	95.5
	M38	10	960	[2, 4, 2, 2, 4, 2, 4, 2, 4, 8]	[3840, 2240, 3328, 3776, 3648, 3200, 2944, 2368, 3968, 2880]	98.8
	M39	10	960	[2, 4, 2, 2, 2, 2, 4, 2, 4, 8]	[3840, 2240, 3328, 3776, 3200, 3200, 3968, 2368, 3968, 2816]	99.8

Table 6: Detailed architectural parameters for all models in Figure 8 with GPT-2 backbone.

		n_{layer}	d_{model}	n_{head}	d_{inner}	DecoderParams (M)
baseline		∈[1,16]	1024	12	3072	-
ARM	M1	2	512	[2, 2]	[1920, 1920]	6.0
	M2	3	320	[8, 2, 4]	[1920, 1920, 3712]	6.1
	M3	2	576	[2, 2]	[1344, 3200]	7.9
	M4	3	384	[2, 8, 2]	[3840, 2368, 3328]	9.1
	M5	5	384	[4, 4, 2, 4, 4]	[2880, 1920, 960, 2496, 1280]	10.3
	M6	2	768	[2, 2]	[1600, 2240]	10.6
	M7	5	320	[4, 2, 2, 4, 2]	[1344, 2240, 3776, 3008, 3648]	11.0
	M8	3	768	[2, 2, 4]	[1856, 1792, 1920]	15.7
	M9	3	704	[2, 2, 2]	[3136, 2112, 1920]	16.1
	M10	2	960	[4, 2]	[3584, 2304]	18.7
	M11	6	448	[4, 4, 2, 2, 4, 2]	[3072, 2112, 4032, 2688, 1600, 3072]	19.7
	M12	3	960	[4, 4, 2]	[2368, 2560, 2048]	24.5
	M13	4	704	[4, 8, 4, 2]	[3008, 3776, 2560, 3648]	26.3
	M14	5	704	[4, 2, 4, 2, 8]	[3584, 3136, 3776, 3072, 1856]	31.7
	M15	3	960	[2, 2, 2]	[3392, 3648, 3840]	32.0
	M16	4	960	[4, 2, 8, 2]	[2048, 3328, 1984, 1856]	32.5
	M17	7	704	[2, 4, 4, 4, 8, 2, 2]	[3008, 2560, 1920, 1856, 2112, 1728, 3136]	36.9
	M18	4	960	[2, 2, 4, 8]	[3392, 3456, 2432, 2304]	37.0
	M19	5	832	[4, 4, 4, 4, 4]	[3840, 1920, 4032, 3072, 3968]	41.9
	M20	5	960	[8, 4, 2, 2, 4]	[2560, 2048, 3648, 1728, 2304]	42.1
	M21	5	960	[4, 4, 2, 2, 2]	[3072, 2240, 1984, 2176, 3520]	43.4
	M22	5	960	[2, 4, 4, 4, 2]	[2496, 3648, 3328, 3392, 2112]	47.2
	M23	6	832	[4, 2, 4, 4, 2, 4]	[2496, 3200, 1664, 3904, 3520, 3840]	47.7
	M24	6	960	[8, 2, 2, 2, 8, 4]	[2304, 3328, 3456, 1856, 1792, 2112]	50.7
	M25	5	960	[4, 8, 2, 4, 4]	[3264, 2688, 4032, 3968, 3712]	52.4
	M26	6	960	[2, 4, 4, 2, 2, 2]	[3008, 2624, 4032, 2688, 3520, 2624]	57.7
	M27	6	960	[2, 4, 4, 2, 8, 2]	[2304, 3648, 3328, 3648, 3904, 1728]	57.8
	M28	6	960	[4, 4, 2, 4, 2, 2]	[3072, 2368, 4032, 4032, 3776, 3264]	61.6
	M29	7	960	[2, 2, 2, 8, 4, 8, 4]	[3008, 2304, 1920, 1984, 3520, 2816, 3712]	62.9
	M30	7	960	[2, 4, 4, 4, 4, 2, 2]	[3200, 4032, 2048, 2624, 2112, 2752, 2880]	63.6
	M31	7	960	[2, 4, 4, 4, 4, 2, 4]	[3584, 3648, 3328, 3392, 3200, 1984, 3200]	68.8
	M32	7	960	[2, 4, 8, 8, 2, 2, 8]	[3008, 3648, 3584, 3648, 3008, 1728, 3712]	68.8
	M33	7	960	[4, 4, 2, 4, 4, 8, 4]	[3584, 3840, 3328, 3392, 3136, 2944, 2496]	69.5
	M34	8	960	[8, 2, 2, 8, 2, 2, 8, 2]	[3008, 3648, 1792, 1984, 3008, 2816, 3712, 3520]	74.7
	M35	8	960	[2, 2, 2, 2, 8, 4, 4, 2]	[3008, 2304, 1792, 3008, 3520, 2880, 3712, 3456]	75.1
	M36	8	960	[2, 2, 2, 2, 2, 2, 4, 8]	[3008, 1792, 3840, 3392, 3520, 3136, 3712, 3520]	79.4
	M37	9	960	[2, 2, 4, 4, 8, 8, 4, 2, 4]	[1664, 1792, 2240, 3904, 3648, 3264, 2176, 3712, 1856]	79.9
	M38	11	832	[8, 4, 2, 4, 4, 2, 8, 4, 4, 8, 8]	[3072, 2368, 4032, 3968, 1664, 3968, 2176, 2624, 3840, 2176, 2112]	83.8
	M39	9	960	[4, 2, 4, 8, 2, 2, 4, 2, 4]	[2496, 3648, 3328, 3392, 3648, 1728, 2880, 3520, 2368]	85.1
	M40	9	960	[4, 2, 4, 8, 4, 2, 4, 2, 4]	[3072, 2816, 4032, 2560, 3648, 1728, 3840, 3264, 3456]	87.8
	M41	10	960	[8, 2, 4, 4, 2, 2, 4, 8, 2, 4]	[3648, 1792, 2432, 1856, 3392, 2304, 3776, 2944, 3136, 3904]	93.0
	M42	10	960	[8, 2, 2, 4, 2, 2, 2, 4, 2, 2]	[3264, 2048, 3520, 3904, 3840, 3840, 2624, 3072, 3776, 2304]	98.8
	M43	12	896	[4, 4, 4, 2, 4, 2, 4, 8, 8, 2, 4, 2]	[2048, 3136, 4032, 1792, 3584, 1728, 3136, 3008, 2560, 3200, 3648, 1728]	98.9
	M44	10	960	[4, 2, 8, 4, 2, 8, 4, 4, 4, 2]	[3584, 3968, 3328, 3904, 2368, 2112, 3904, 3520, 3328, 2688]	99.8
	M45	10	960	[8, 2, 4, 4, 4, 4, 4, 2, 2, 8]	[2688, 3200, 3840, 3392, 3520, 3136, 3392, 3520, 2880, 3200]	99.9
Corei7	M1	2	384	[2, 2]	[3840, 2432]	6.0
	M2	3	320	[2, 2, 2]	[2176, 3072, 2496]	6.2
	M3	2	512	[2, 2]	[1408, 2624]	6.2
	M4	3	384	[2, 2, 2]	[3264, 3456, 3584]	9.7
	M5	2	576	[2, 2]	[3136, 3648]	10.5
	M6	3	448	[2, 2, 2]	[4032, 3648, 4032]	12.9
	M7	4	448	[2, 2, 4, 4]	[3072, 3648, 4032, 1792]	14.5
	M8	2	768	[2, 2]	[3968, 3328]	15.9
	M9	4	576	[2, 2, 2, 2]	[3072, 2752, 3456, 3136]	19.6
	M10	2	960	[2, 2]	[3840, 3264]	21.0
	M11	4	640	[2, 2, 2, 2]	[2176, 3648, 3584, 1920]	21.1
	M12	3	960	[2, 2, 2]	[2176, 3264, 2432]	26.2
	M13	4	768	[2, 2, 2, 2]	[3584, 2112, 3392, 1920]	26.4
	M14	4	768	[2, 2, 2, 2]	[3584, 2560, 3776, 1536]	27.1
	M15	4	832	[2, 2, 2, 2]	[3904, 1984, 3392, 3136]	31.8
	M16	3	960	[2, 2, 2]	[3968, 4032, 2880]	32.0
	M17	5	768	[2, 2, 4, 2, 2]	[3648, 3072, 3392, 1984, 2944]	34.9
	M18	4	960	[2, 2, 2, 2]	[3136, 1984, 3392, 2944]	36.8
	M19	4	960	[2, 2, 2, 4]	[3968, 3456, 3584, 3136]	42.0
	M20	6	768	[4, 2, 2, 4, 2, 4]	[3584, 2112, 3456, 3136, 3840, 2560]	42.9
	M21	7	768	[2, 4, 2, 4, 4, 4, 2]	[2624, 1984, 2496, 3968, 2880, 2112, 4032]	47.5
	M22	5	960	[2, 2, 4, 2, 4]	[2176, 3264, 3392, 3008, 3328]	47.6
	M23	6	960	[4, 4, 2, 4, 2, 2]	[2048, 2624, 3520, 1984, 2880, 2624]	52.3
	M24	6	960	[2, 4, 4, 4, 2, 2]	[1792, 3456, 2752, 2240, 1664, 3840]	52.4
	M25	6	960	[4, 2, 2, 2, 4, 4]	[2176, 1664, 3648, 3136, 3968, 3904]	57.7
	M26	7	960	[2, 2, 4, 4, 2, 2, 8]	[2816, 1792, 3968, 1728, 1664, 3328, 2944]	60.9
	M27	7	896	[2, 2, 4, 2, 2, 2, 2]	[3904, 3264, 3328, 3968, 1728, 2624, 4032]	63.5
	M28	7	960	[4, 2, 4, 2, 2, 2, 2]	[3584, 2560, 1792, 1920, 3968, 2112, 3968]	64.1
	M29	8	960	[2, 2, 2, 4, 2, 2, 2, 4]	[3328, 2432, 2624, 2752, 1664, 2240, 2304, 2816]	68.3
	M30	7	960	[4, 2, 4, 2, 2, 2, 2]	[3904, 2304, 2368, 3584, 3264, 2880, 3904]	68.5
	M31	8	960	[4, 2, 4, 2, 2, 4, 2, 4]	[2560, 3648, 2624, 2112, 3328, 2112, 1792, 3328]	70.9
	M32	8	960	[4, 4, 4, 2, 2, 4, 2, 4]	[2560, 2304, 2624, 4032, 2688, 2624, 3840, 2816]	74.7
	M33	9	960	[2, 4, 2, 4, 2, 4, 2, 2, 4]	[3072, 3264, 2944, 1984, 2880, 3520, 2112, 2624, 1728]	79.6
	M34	10	896	[2, 2, 4, 2, 2, 2, 2, 2, 4, 2]	[2816, 3264, 3584, 1792, 3136, 3584, 2240, 2240, 1920, 2752]	81.2
	M35	9	960	[8, 2, 2, 2, 4, 4, 2, 4, 4]	[3904, 3648, 2432, 3136, 3264, 2816, 2240, 3072, 3840]	87.7
	M36	10	960	[4, 4, 2, 2, 4, 4, 2, 4, 4, 2]	[2176, 3264, 2752, 3136, 3968, 3520, 3776, 3328, 1728, 2496]	94.9
	M37	10	960	[4, 2, 4, 2, 2, 2, 2, 4, 2, 2]	[3904, 2112, 2496, 3968, 3968, 2624, 3904, 2304, 3200, 3840]	99.0
	M38	11	960	[4, 2, 2, 4, 2, 4, 2, 2, 4, 4, 4]	[2176, 4032, 3264, 3840, 2688, 1984, 1728, 2944, 1920, 2368, 3840]	99.8