1a)

Number of layers (N)	Parameters (d_{model}=512)	Parameters (d_{model}=1024)	Parameters (d_{model}=2048)
1	2102784	6301696	20990976
2	4205568	12603392	41981952
3	6308352	18905088	62972928
4	8411136	25206784	83963904
5	10513920	31508480	104954880
6	12616704	37810176	125945856
7	14719488	44111872	146936832
8	16822272	50413568	167927808
9	18925056	56715264	188918784
10	21027840	63016960	209909760
11	23130624	69318656	230900736
12	25233408	75620352	251891712

An increase in the number of layers, **N**, leads to a linear increase in the number of total number of parameters. This means when N = 1, total parameters = 2102784, and when N =2, total parameters = (2102784) * 2 and so on.

An increase in the token representation size of d_{model} causes a non-linear increase in parameters. When d_{model} size is multiplied by 2, the total parameters multiply approximately by 3. When d_{model} size is multiplied by 4, the total parameters multiply approximately by 10.

b)

"Multi-head attention works as an ensemble of heads in the transformer architecture."

I agree with the statement. The data is logically and uniformly split across the attention heads. This means that the data is not actually physically split, but it just has each head operating on a section of the data. The computations for **N** heads are carried out by a single matrix operation rather than **N** separate operations. Therefore, multi-head attention is carried out together in parallel, thus working as an ensemble.

c)

In transformers, using attention allows the decoder to gather information from about the input sequence from the encoder's hidden states based on the current state. This allows the decoder to learn more about the nuances and dependencies between words. Furthermore, the distance between words is not a problem in transformers as compared to LSTM/GRU, which is unable to capture the dependencies properly if the words are too far apart in a sentence.

Transformers also does computation in parallel, which is faster than LSTM/GRU which does it in sequential order. On the other hand, transformers need to have position encoding in order to remember the sequence of the words in a sentence, where LSTM/GRU have no need for that since they process the data sequentially.

For decoder inference, transformer feeds the entire output sequence from previous timestep back to the decoder, whereas LSTM/GRU only feeds the last word.

d)

During conversion of the input and target sequence into embeddings, the encoder and decoder in transformer also have a position encoding layer that computes the position encoding independently of the input sequence. The position encoding are fixed values that captures the sequence information, and the values depend on the max length of the sequence.

e)

Table of hyper-parameters tested; unlisted values are identical to those of the base model.

	N	d_{model}	h	ffn_dim	development set perplexity	test perplexity
Base →	6	512	8	1024	173.63	156.99
			1		161.13	149.00
			2		158.68	146.05
			4		156.97	143.71
			16		155.79	144.4
	1				232.73	211.22
	2				180.23	165.12
	3				184.13	170.51
	4				161.24	149.12
	5				178.56	157.67
	7				153.67	142.02
	8				154.14	142.78
	9				176.98	160.05
	10				992.81	924.15
	11				973.59	916.62
	12				971.89	913.60
		1024			161.68	149.16
		2048			172.73	159.86
				512	157.08	145.94
Best development set perplexity →				2048	143.29	155.11
	7		4		175.30	160.73
	8		4		173.68	156.40
	7	2048			176.58	163.81
	7	1024			158.86	144.79
	8	1024			158.95	145.20
	7			512	174.23	159.43
Best test set perplexity →	7			2048	152.63	141.49
	8			512	173.23	159.60
	8			2048	153.58	142.46
	7	1024	4	512	160.18	147.97
	8	1024	4	512	156.73	145.51
	7	1024	16	512	180.42	165.79
	8	1024	16	512	158.74	146.27

The best model based on development set perplexity is:

N	d_{model}	h	ffn_dim	development set perplexity	test perplexity
6	512	8	2048	143.29	155.11

References

Doshi K. (Dec 2020). Transformers Explained Visually (Part 1): Overview of Functionality Retrieved from https://towardsdatascience.com/transformers-explained-visually-part-1-overview-of-functionality-95a6dd460452

Doshi K. (Jan 2021). Transformers Explained Visually (Part 2): How it works, step-by-step Retrieved from https://towardsdatascience.com/transformers-explained-visually-part-2-how-it-works-step-by-step-b49fa4a64f34

Doshi K. (Jan 2021). Transformers Explained Visually (Part 3): Multi-head Attention, deep dive Retrieved from https://towardsdatascience.com/transformers-explained-visually-part-3-multi-head-attention-deep-dive-1c1ff1024853

Thiruvengadam A. (Oct 2018). Retrieved from https://medium.com/@adityathiruvengadam/transformer-architecture-attention-is-all-you-need-aeccd9f50d09