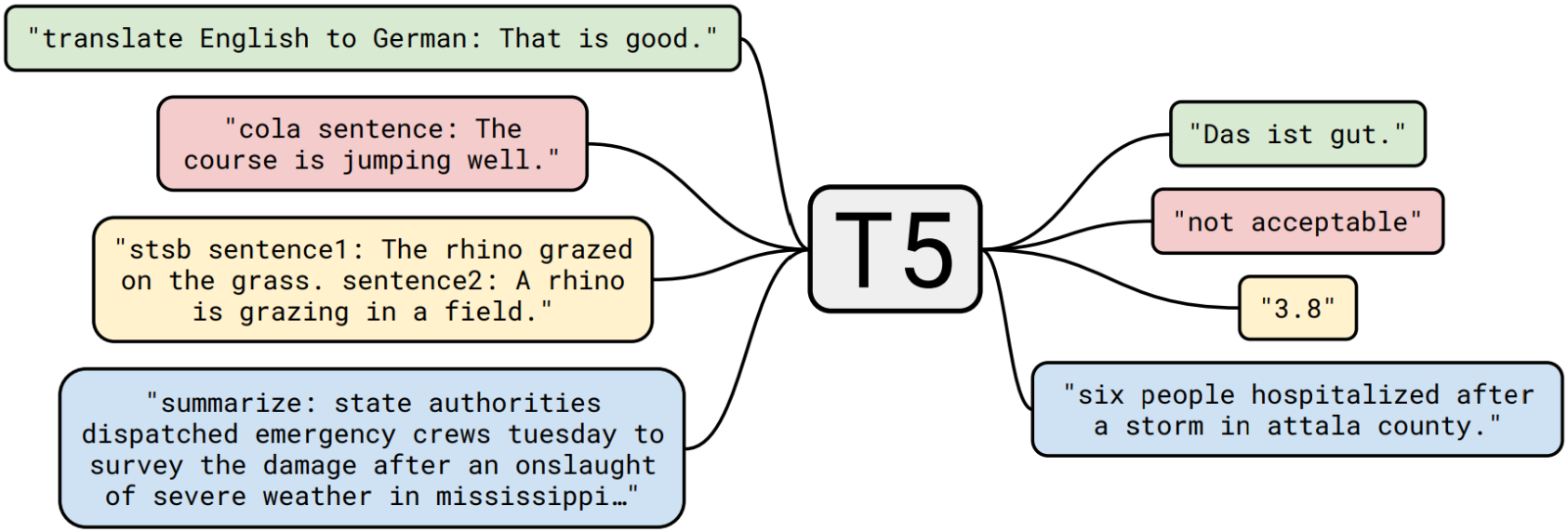
Using the Transformer

Text-to-Text Transfer Transformer

Learning goals

- shortcomings of BERT & Co.
- everything as text-to-text
- Dynamic Masking

REVISITING TEXT-TO-TEXT TASKS



▸ Source: Raffel et al., 2020

Ingredients:

- `<task prefix>` for each task

  → Concatenation with the input
- Output label / score formatted as string

TEXT-TO-TEXT TRANSFER TRANSFORMER

Short summary:

- In short: T5 (bc of the five Ts)
- A complete encoder-decoder Transformer architecture

  Question: Why is this important?
- Relative positional embeddings
- All tasks reformulated as text-to-text tasks

  $\rightarrow$ Probably the most important innovation of this work
- From BERT-size $\times 2$ (since enc-dec) up to 11B parameters
- Paradigm shift from *Sequential Transfer Learning* towards true *Multi-Task Learning*

▸ Nice Animation (similar to figure before)

INPUT AND OUTPUT FORMAT

Input Side:

- SentencePiece framework w/ WordPiece tokens
- Add task-specific (text) prefix to the original input
- Choice of the prefix: Hyperparameter!!

$\rightarrow$ Changing this had limited impact

$\rightarrow$ No extensive experiments performed by the authors

Output Side:

- Output as a word or a piece of text (also numerical similarity scores)
- If output not present in set of potential alternatives, prediction considered as wrong (many other possible ways; this is not trivial)

TASK PREFIX VS. PROMPTING

Adding task-specific (text) prefix:

- Add task-specific (text) prefix to the original input
- Model is fine-tuned on samples prepended with this prefix

$\rightarrow$ It learns to recognize what it is expected to do when encountering a specific prefix at test time

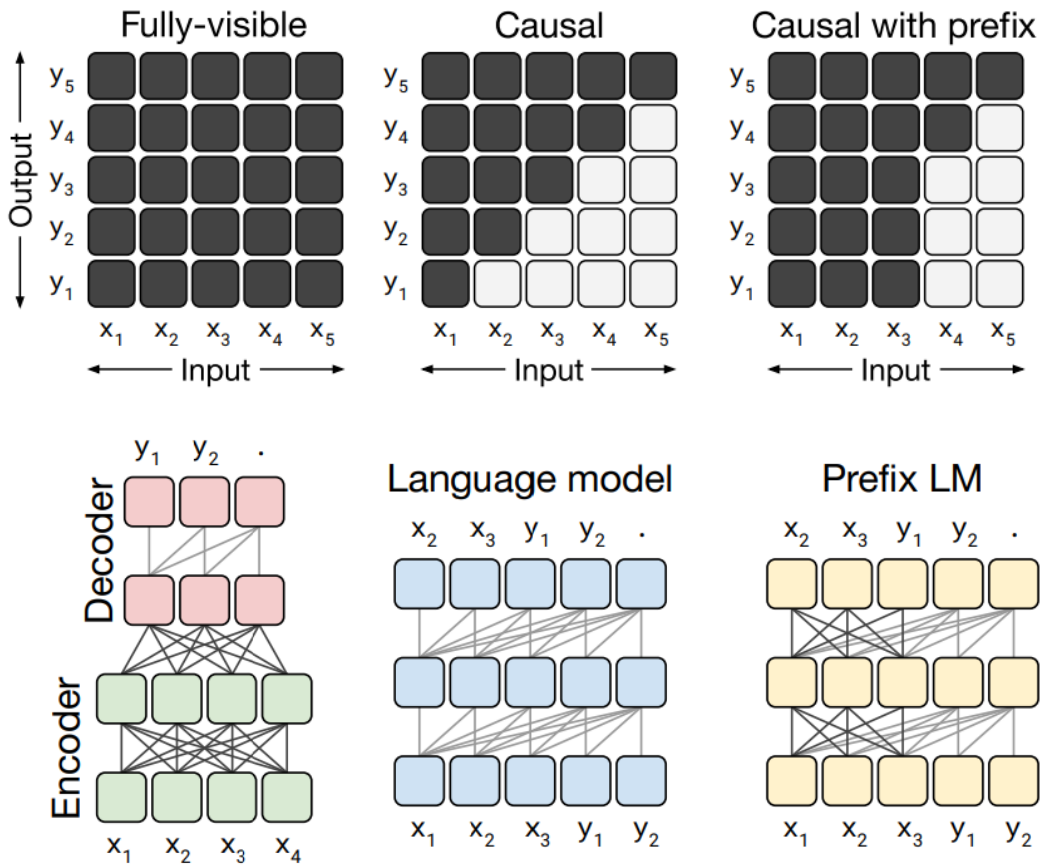
$\rightarrow$ Probably because of this: limited impact found by the authors

Prompting:

- Refers to just querying a trained w/o fine-tuning it (cf. next chapter)
- Paradigm of Few-/Zero-Shot Learning
- This is found to have a *huge* impact on model performance

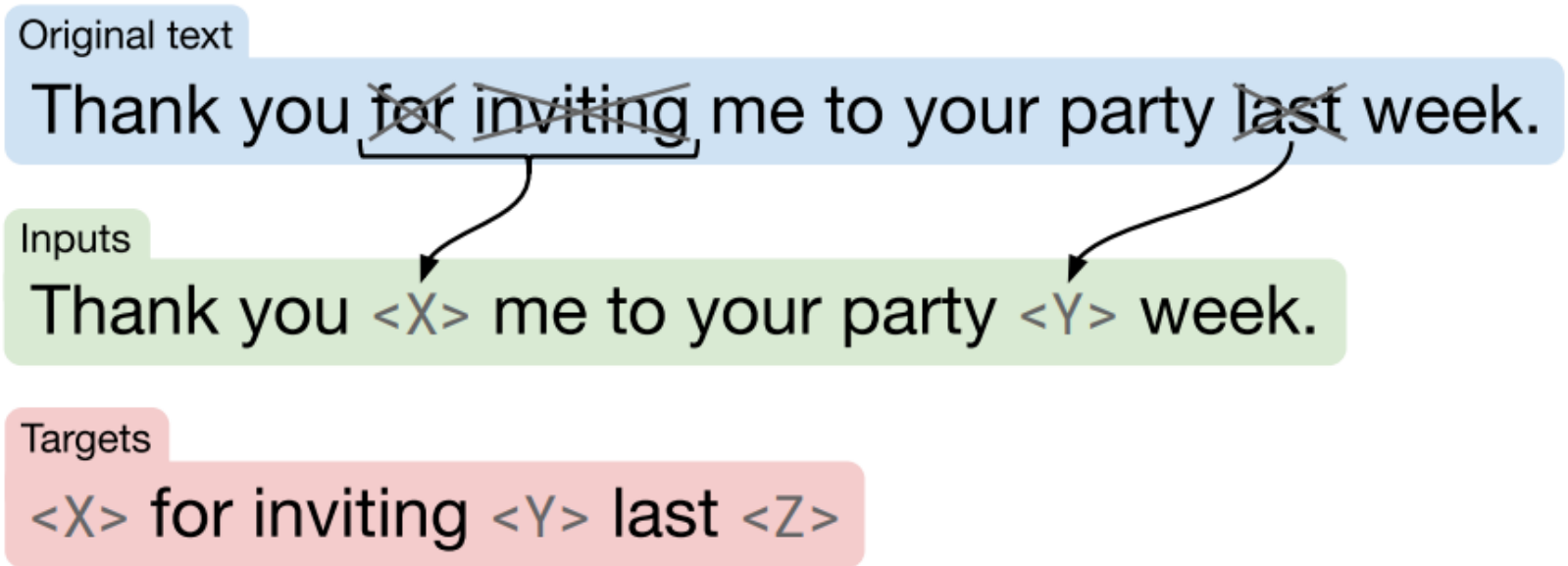
ARCHITECTURAL DIFFERENCES

Attention patterns (top) and Schematics of considered architectures (bottom)



▸ Source: Raffel et al., 2020

PRE-TRAINING OBJECTIVE



▸ Source: Raffel et al., 2020

1. Mark spans in input sequence for corruptions
2. Replace tokens with sentinel tokens
3. Predict sentinel tokens and replaced tokens

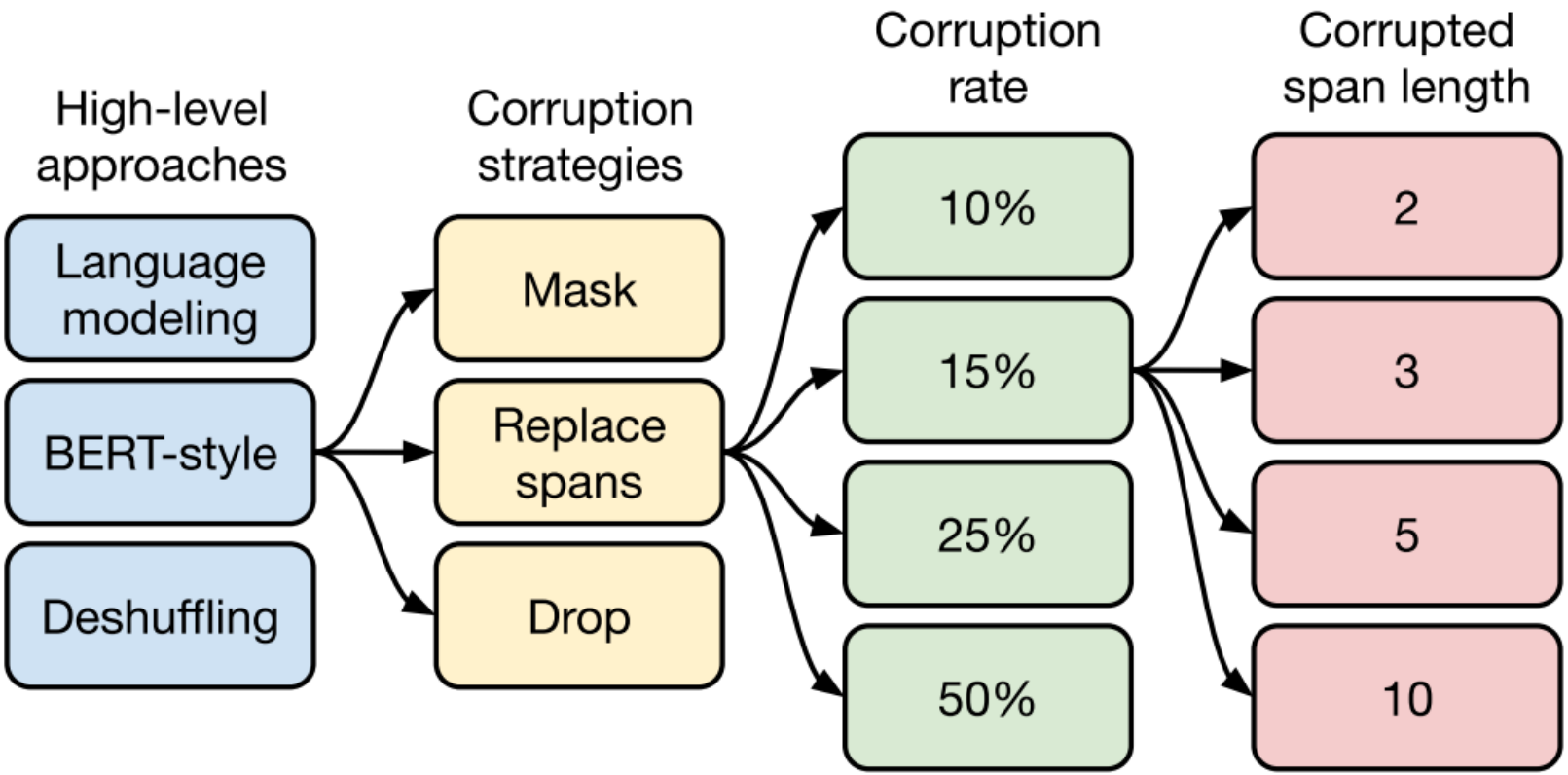
PRE-TRAINING OBJECTIVES

- Authors experimented with different objectives
- Most of them rely in some way on MLM

Objective	Inputs	Targets
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style Devlin et al. (2018)	Thank you <M> <M> me to your party apple week .	*(original text)*
Deshuffling	party me for your to . last fun you inviting week Thank	*(original text)*
MASS-style Song et al. (2019)	Thank you <M> <M> me to your party <M> week .	*(original text)*
I.i.d. noise, replace spans	Thank you <X> me to your party <Y> week .	<X> for inviting <Y> last <Z>
I.i.d. noise, drop tokens	Thank you me to your party week .	for inviting last
Random spans	Thank you <X> to <Y> week .	<X> for inviting me <Y> your party last <Z>

▸ Source: Raffel et al., 2020

PRE-TRAINING OBJECTIVES



▸ Source: Raffel et al., 2020

THE COLOSSAL CLEAN CRAWLED CORPUS (C4)

- Effort to measure the effect of quality, characteristics & size of the pre-training resources
- Common Crawl as basis, careful cleaning and filtering for English language
- Orders of magnitude larger (750GB) compared to commonly used corpora

THE COLOSSAL CLEAN CRAWLED CORPUS (C4)

Experiments (with respect to C4)

Data set	Size	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ C4	745GB	83.28	**19.24**	80.88	71.36	**26.98**	**39.82**	**27.65**
C4, unfiltered	6.1TB	81.46	19.14	78.78	68.04	26.55	39.34	27.21
RealNews-like	35GB	**83.83**	**19.23**	80.39	72.38	**26.75**	**39.90**	**27.48**
WebText-like	17GB	**84.03**	**19.31**	**81.42**	71.40	**26.80**	**39.74**	**27.59**
Wikipedia	16GB	81.85	**19.31**	81.29	68.01	**26.94**	39.69	**27.67**
Wikipedia + TBC	20GB	83.65	**19.28**	**82.08**	**73.24**	**26.77**	39.63	**27.57**

Number of tokens	Repeats	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Full data set	0	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
2^{29}	64	**82.87**	**19.19**	**80.97**	**72.03**	**26.83**	**39.74**	**27.63**
2^{27}	256	82.62	**19.20**	79.78	69.97	**27.02**	**39.71**	27.33
2^{25}	1,024	79.55	18.57	76.27	64.76	26.38	39.56	26.80
2^{23}	4,096	76.34	18.33	70.92	59.29	26.37	38.84	25.81

▸ Source: Raffel et al., 2020

T5 - EXHAUSTIVE EXPERIMENTS

Performed experiments with respect to ..

- *.. architecture, size & objective*

→ enc, dec, enc-dec

→ Between 60M and 11B parameters

- *.. details of the Denoising objective (which was found to work best)*
- *.. fine-tuning methods*

→ Adapter layers

→ Gradual Unfreezing (cf. ULMFiT)

- *.. Multi-task learning strategies*

→ Examples-proportional mixing

→ Temperature-scaled mixing

BENCHMARK RESULTS

Model	GLUE Average	CoLA Matthew's	SST-2 Accuracy	MRPC F1	MRPC Accuracy	STS-B Pearson	STS-B Spearman
Previous best	89.4[a]	69.2[b]	97.1[a]	**93.6**[b]	**91.5**[b]	92.7[b]	92.3[b]
T5-Small	77.4	41.0	91.8	89.7	86.6	85.6	85.0
T5-Base	82.7	51.1	95.2	90.7	87.5	89.4	88.6
T5-Large	86.4	61.2	96.3	92.4	89.9	89.9	89.2
T5-3B	88.5	67.1	97.4	92.5	90.0	90.6	89.8
T5-11B	**90.3**	**71.6**	**97.5**	92.8	90.4	**93.1**	**92.8**

Model	QQP F1	QQP Accuracy	MNLI-m Accuracy	MNLI-mm Accuracy	QNLI Accuracy	RTE Accuracy	WNLI Accuracy
Previous best	74.8[c]	**90.7**[b]	91.3[a]	91.0[a]	**99.2**[a]	89.2[a]	91.8[a]
T5-Small	70.0	88.0	82.4	82.3	90.3	69.9	69.2
T5-Base	72.6	89.4	87.1	86.2	93.7	80.1	78.8
T5-Large	73.9	89.9	89.9	89.6	94.8	87.2	85.6
T5-3B	74.4	89.7	91.4	91.2	96.3	91.1	89.7
T5-11B	**75.1**	90.6	**92.2**	**91.9**	96.9	**92.8**	**94.5**

▸ Source: Raffel et al., 2020

BENCHMARK RESULTS

Model	SQuAD EM	SQuAD F1	SuperGLUE Average	BoolQ Accuracy	CB F1	CB Accuracy	COPA Accuracy
Previous best	90.1[a]	95.5[a]	84.6[d]	87.1[d]	90.5[d]	95.2[d]	90.6[d]
T5-Small	79.10	87.24	63.3	76.4	56.9	81.6	46.0
T5-Base	85.44	92.08	76.2	81.4	86.2	94.0	71.2
T5-Large	86.66	93.79	82.3	85.4	91.6	94.8	83.4
T5-3B	88.53	94.95	86.4	89.9	90.3	94.4	92.0
T5-11B	**91.26**	**96.22**	**88.9**	**91.2**	**93.9**	**96.8**	**94.8**

Model	MultiRC F1a	MultiRC EM	ReCoRD F1	ReCoRD Accuracy	RTE Accuracy	WiC Accuracy	WSC Accuracy
Previous best	84.4[d]	52.5[d]	90.6[d]	90.0[d]	88.2[d]	69.9[d]	89.0[d]
T5-Small	69.3	26.3	56.3	55.4	73.3	66.9	70.5
T5-Base	79.7	43.1	75.0	74.2	81.5	68.3	80.8
T5-Large	83.3	50.7	86.8	85.9	87.8	69.3	86.3
T5-3B	86.8	58.3	91.2	90.4	90.7	72.1	90.4
T5-11B	**88.1**	**63.3**	**94.1**	**93.4**	**92.5**	**76.9**	**93.8**

▸ Source: Raffel et al., 2020

BENCHMARK RESULTS

Model	WMT EnDe BLEU	WMT EnFr BLEU	WMT EnRo BLEU	CNN/DM ROUGE-1	CNN/DM ROUGE-2	CNN/DM ROUGE-L
Previous best	**33.8**[e]	**43.8**[e]	**38.5**[f]	43.47[g]	20.30[g]	40.63[g]
T5-Small	26.7	36.0	26.8	41.12	19.56	38.35
T5-Base	30.9	41.2	28.0	42.05	20.34	39.40
T5-Large	32.0	41.5	28.1	42.50	20.68	39.75
T5-3B	31.8	42.6	28.2	42.72	21.02	39.94
T5-11B	32.1	43.4	28.1	**43.52**	**21.55**	**40.69**

▸ Source: Raffel et al., 2020

T5 - EXHAUSTIVE EXPERIMENTS

Conclusions

- Encoder-decoder architecture works best in this "text-to-text" setting
- More data, larger models & ensembling all boost the performance
  - Larger models trained for fewer steps better than smaller models on more data
  - Ensembling: Using same base pre-trained models worse than complete separate model ensembles
- Different denoising objectives work similarly well
- Updating *all* model parameters during fine-tuning works best (but expensive)

DIFFERENT T5 VERSIONS

Similar to BERT T5 has different versions, which all have a different parameter count ▸ Raffel et al., 2020

Model	Parameters	n_{layers}	d_{model}	d_{FFN}	d_{qkv}	n_{heads}
small	60M	6	512	2048	64	8
base	220M	12	768	3072	64	12
large	770M	24	1024	4096	64	16
3B	3B	24	1024	16384	128	32
11B	11B	24	1024	65536	128	128

- For T5 small, base and large we have the following parameter count per encoder/decoder layer:
  - $N_{Encoder} = 12 \cdot d_{model}^2$
  - $N_{Decoder} = 16 \cdot d_{model}^2$

For T5 3B and 11B $N_{Encoder}$ and $N_{Decoder}$ are different!

APPROXIMATE PARAMETER COUNT (1)

The general formula is:

$$N_{Total} = n_{layers} \cdot (N_{Encoder} + N_{Decoder}) + N_{Embedding}$$

- Let $V = 32000$ be the size of the vocabulary, we then have:

$$N_{Embedding} = 32000 \times d_{model}$$

- **T5 small**:

$$N_{small} = 6 \cdot (12 \cdot 512^2 + 16 \cdot 512^2) + 32000 \times 512 = 60,424,192$$

- **T5 base**:

$$N_{base} = 12 \cdot (12 \cdot 768^2 + 16 \cdot 768^2) + 32000 \times 768 = 222,756,864$$

- **T5 large**:

$$N_{large} = 24 \cdot (12 \cdot 1024^2 + 16 \cdot 1024^2) + 32000 \times 1024 = 737,411,072$$

REVISITING: MHA AND FFN (1)

- In the transformer chapter we derived the FFN parameter count to be $\mathbf{8 \cdot d_{model}}$
- We also derived the MHA parameter count to be $\mathbf{4 \cdot d_{model}^2}$
- But this is only the case if we choose $d_{FFN} = 4 \cdot d_{model}$ and $d_{qkv} = \frac{d_{model}}{n_{heads}}$, with d_{FFN} being the hidden dimension of the FFN and d_{qkv} being the embedding dimension of each head in MHA
- For the two bigger model variants we have the following relations:
  - **T5 3B**: $d_{FFN} = 16 \cdot d_{model}$ and $d_{qkv} \neq \frac{d_{model}}{n_{heads}}$
  - **T5 11B**: $d_{FFN} = 64 \cdot d_{model}$ and $d_{qkv} \neq \frac{d_{model}}{n_{heads}}$
- Thats why we can't use the same formulas as for the three previous variants

REVISITING: MHA AND FFN (2)

- In MHA $n_{heads} \cdot d_{qkv}$ is no longer d_{model}:

$$\begin{aligned} N_{attention} &= n_{heads} \cdot d_{qkv} \times d_{model} + (d_{model} \times d_{qkv}) \times n_{heads} \cdot 3 \\ &= 4 \cdot n_{heads} \cdot d_{qkv} \times d_{model}; \quad \text{instead of:} \quad 4 \cdot d_{model}^2 \end{aligned}$$

- For the FFN d_{FFN} is no longer $4 \cdot d_{model}$:

$$\begin{aligned} N_{FFN} &= d_{model} \times d_{FFN} + d_{FFN} \times d_{model} \\ &= 2 \cdot d_{model} \times d_{FFN}; \quad \text{instead of:} \quad 8 \cdot d_{model}^2 \end{aligned}$$

- For the Encoder and Decoder that means:
  - $N_{Encoder} = 4 \cdot n_{heads} \cdot d_{qkv} \times d_{model} + 2 \cdot d_{model} \times d_{FFN}$
  - $N_{Decoder} = 2 \cdot 4 \cdot n_{heads} \cdot d_{qkv} \times d_{model} + 2 \cdot d_{model} \times d_{FFN}$

APPROXIMATE PARAMETER COUNT (2)

The general formula is still:

$$N_{Total} = n_{layers} \cdot (N_{Encoder} + N_{Decoder}) + N_{Embedding}$$

- **T5 3B**:

$$\begin{aligned} N_{3B} &= 24 \cdot ([4 \cdot 32 \cdot 128 \times 1024 + 2 \cdot 1024 \times 16384] \\ &\quad + [2 \cdot 4 \cdot 32 \cdot 128 \times 1024 + 2 \cdot 1024 \times 16384]) + 32000 \cdot 1024 \\ &= 2,851,340,288 \approx 3B \end{aligned}$$

- **T5 11B**:

$$\begin{aligned} N_{11B} &= 24 \cdot ([4 \cdot 128 \cdot 128 \times 1024 + 2 \cdot 1024 \times 65536] \\ &\quad + [2 \cdot 4 \cdot 128 \cdot 128 \times 1024 + 2 \cdot 1024 \times 65536]) + 32000 \cdot 1024 \\ &= 11,307,057,152 \approx 11B \end{aligned}$$