T5
Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

Background

Pre-train → Fine-tune

Goal:

explore how different factor will affect the performance.

- pre-training objectives
- architectures
- unlabeled datasets
- transfer approaches

Unified Text-to-Text View

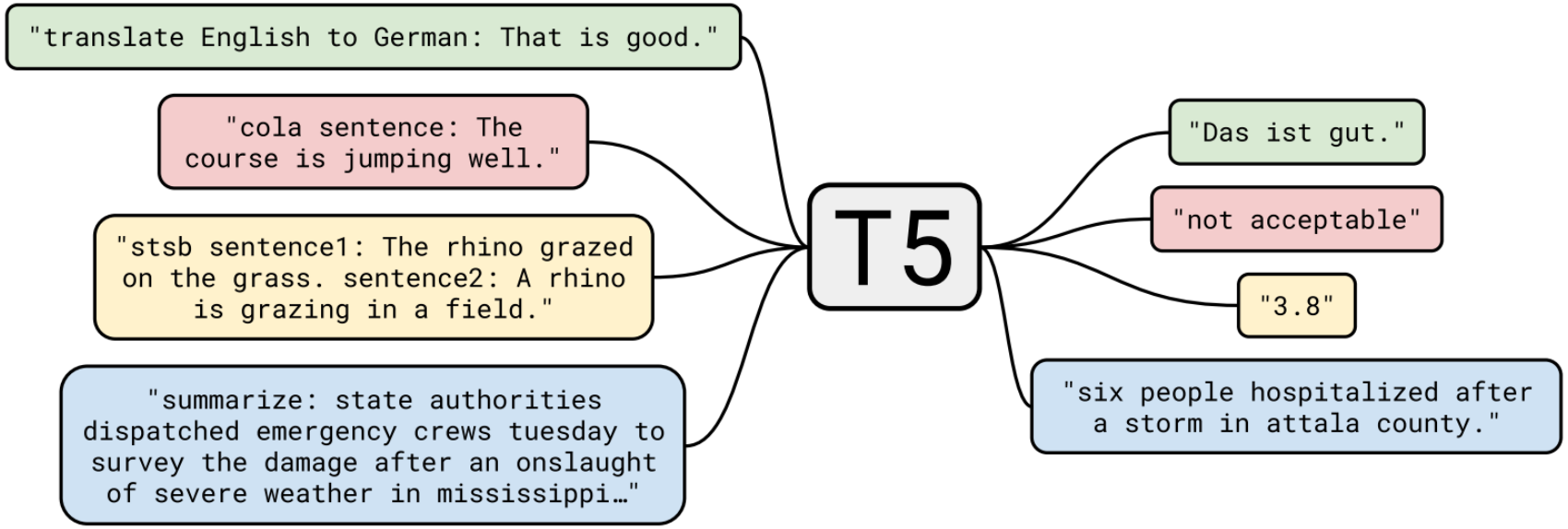


Figure 1: A diagram of our text-to-text framework. Every task we consider – including translation, question answering, and classification – is cast as feeding our model text as input and training it to generate some target text. This allows us to use the same model, loss function, hyperparameters, etc. across our diverse set of tasks. It also provides a standard testbed for the methods included in our empirical survey. "T5" refers to our model, which we dub the "**T**ext-**t**o-**T**ext **T**ransfer **T**ransformer".

Model details

- Encoder-decoder Transformer
- Relative Positional Self-Attention

$$\text{RelativeAttention} = \text{Softmax}\left(\frac{QK^\top + S^{rel}}{\sqrt{D_h}}\right)V.$$

- S^{rel} is the simplified positional embedding

the offset between the "key" and "query" being compared in the self-attention mechanism. We use a simplified form of position embeddings where each "embedding" is simply a scalar that is added to the corresponding logit used for computing the attention weights. For efficiency, we also share the position embedding parameters across all layers in our model, though within a given layer each attention head uses a different learned position embedding. Typically, a fixed number of embeddings are learned, each corresponding to a range of possible key-query offsets. In this work, we use 32 embeddings for all of our models with ranges that increase in size logarithmically up to an offset of 128 beyond which we assign all relative positions to the same embedding. Note that a given layer is insensitive to relative position beyond 128 tokens, but subsequent layers can build a sensitivity to larger offsets by combining local information from previous layers.

Dataset: Colossal Clean Crawled Corpus

- Goal: the effect of the *quality*, *characteristics*, and *size* of unlabeled data
- Source: Common Crawl (20TB/month, noisy)
- Data Cleaning:

  (Heuristics)
- 750G

- We only retained lines that ended in a terminal punctuation mark (i.e. a period, exclamation mark, question mark, or end quotation mark).
- We removed any page that contained any word on the "List of Dirty, Naughty, Obscene or Otherwise Bad Words".[6]
- Many of the scraped pages contained warnings stating that Javascript should be enabled so we removed any line with the word Javascript.
- Some pages had placeholder "lorem ipsum" text; we removed any page where the phrase "lorem ipsum" appeared.
- Some pages inadvertently contained code. Since the curly bracket "{" appears in many programming languages (such as Javascript, widely used on the web) but not in natural text, we removed any pages that contained a curly bracket.
- To deduplicate the dataset, we discarded all but one of any three-sentence span occurring more than once in the dataset.

Additionally, since most of our downstream tasks are focused on English-language text, we used `langdetect`[7] to filter out any pages that were not classified as English with a probability of at least 0.99.

Downstream Tasks

- Text classification: GLUE and SuperGLUE
- Abstractive summarization: CNN/Daily Mail
- QA: SQuAD
- Translation: WMT English to German, French, and Romanian

Input & Output

- “text-to-text” format
- consistent training objective: maximum likelihood
- task-specific (text) prefix

- Mismatch label Issue

word “entailment”. Note that an issue arises if our model outputs text on a text classification task that does not correspond to any of the possible labels (for example if the model outputs “hamburger” when the only possible labels for a task were “entailment”, “neutral”, or “contradiction”). In this case, we always count the model’s output as wrong, though we never observed this behavior in any of our trained models. A diagram of our text-to-text framework with a few input/output examples is

Input & Output

- Regression Task:
  - Convert to 21-class classification

Following this approach allows us to straightforwardly use a text-to-text format for every task except STS-B, which is a regression task where the goal is to predict a similarity score between 1 and 5. We found that most of these scores were annotated in increments of 0.2, so we simply rounded any score to the nearest increment of 0.2 and converted the result to a literal string representation of the number (e.g. the floating-point value 2.57 would be mapped to the string "2.6"). At test time, if the model outputs a string corresponding to a number between 1 and 5, we convert it to a floating-point value; otherwise, we treat the model's prediction as incorrect. This effectively recasts the STS-B regression problem as a 21-class classification problem.

Input & Output

- Winograd Task (ambiguation):
  - highlighting

the passage might be "The city councilmen refused the demonstrators a permit because they feared violence.", which contains the ambiguous pronoun "they" that could refer to "city councilmen" or "demonstrators". We cast the WNLI, WSC, and DPR tasks as text-to-text problems by highlighting the ambiguous pronoun in the text passage and asking the model to predict the noun that it refers to. The example mentioned above would be transformed to the input "The city councilmen refused the demonstrators a permit because *they* feared violence." and the model would be trained to predict the target text "The city councilmen".

Empirical Survey

Methodology “coordinate descent”

Baseline → Architecture → Objective → Dataset

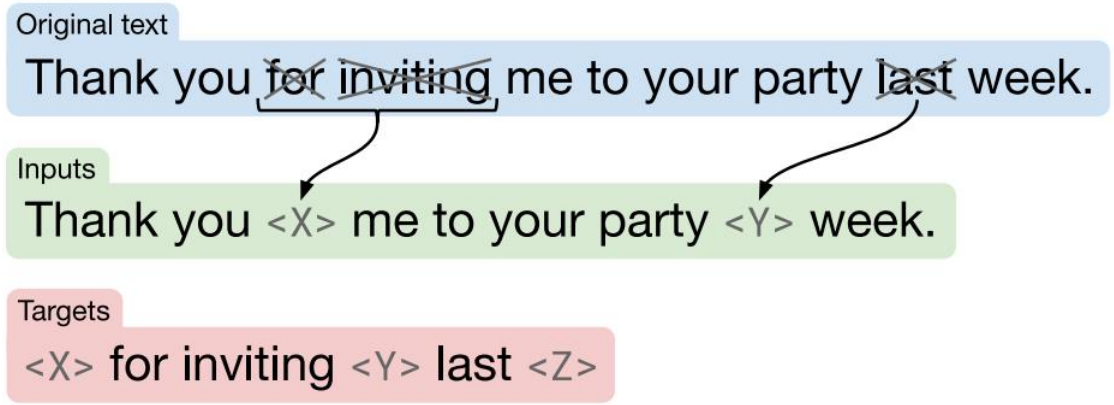
→ Transfer Approach → Scaling

Baseline

- Encoder-Decoder Transformer
- Denoising objective
- BERT-base Size Encoder and Decoder (2x larger)
- Multilingual Vocabulary
  - 32,000 word pieces
  - SentencePiece

Baseline

- Denoising objective



- Drop 15% tokens

Baseline Results

	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline average	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Baseline standard deviation	0.235	0.065	0.343	0.416	0.112	0.090	0.108
No pre-training	66.22	17.60	50.31	53.04	25.86	**39.77**	24.04

Table 1: Average and standard deviation of scores achieved by our baseline model and training procedure. For comparison, we also report performance when training on each task from scratch (i.e. without any pre-training) for the same number of steps used to fine-tune the baseline model. All scores in this table (and every table in our paper except Table 14) are reported on the validation sets of each dataset.

Comparable to BERT-base

Baseline Details (Pre-train)

- AdaFactor
- Dropout: 0.1
- Max length: 512
- Batch Size: 128
  - pack multiple sentence into one sample: [1 1 1 1 1 1 0 0 2 2 2 0 0 3 3 3 3 3 0 0 4 4 4]
- 34B tokens << BERT (137B) <<RoBERTa (2.2T)
- “inverse square root” Learning Rate
  - triangular is better but not comparable
- 10000 warmup

Baseline Details (Fine-tune)

- 2^{18} steps
- constant learning rate: 0.001
- Batch Size: 128
- Length: 512
- 5,000 steps/checkpoint

Empirical Survey

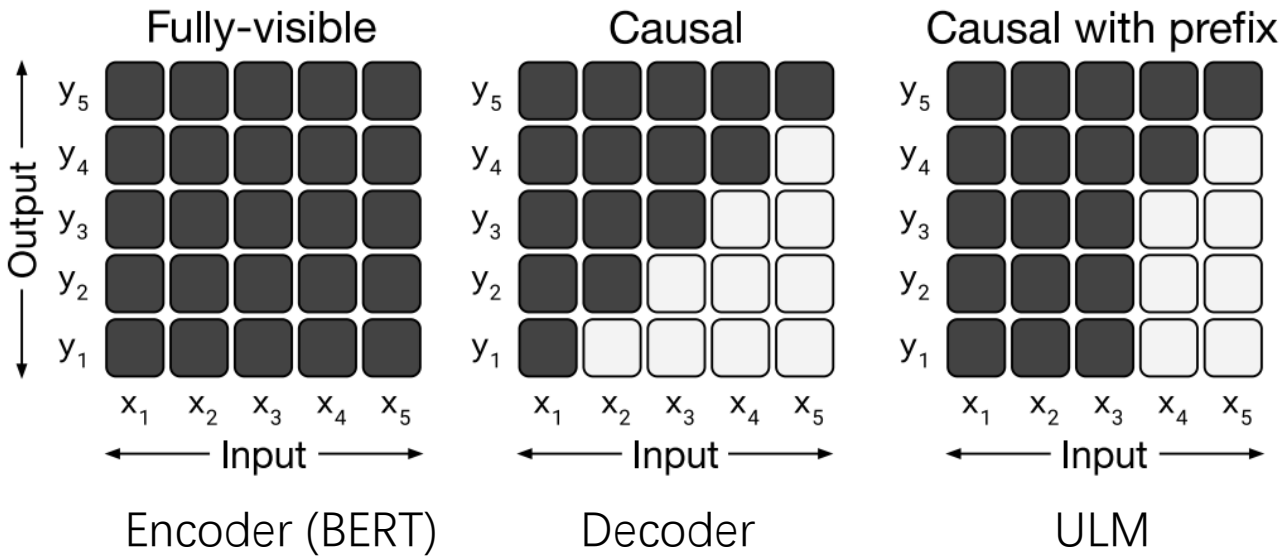
Methodology “coordinate descent”

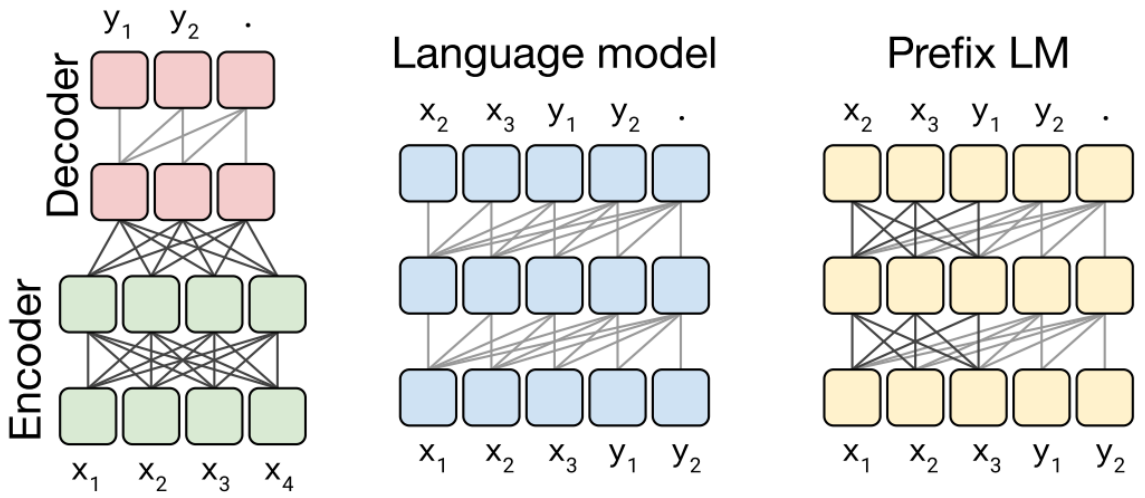
Baseline → **Architecture** → Objective → Dataset

→ Transfer Approach → Scaling

Model Architectures

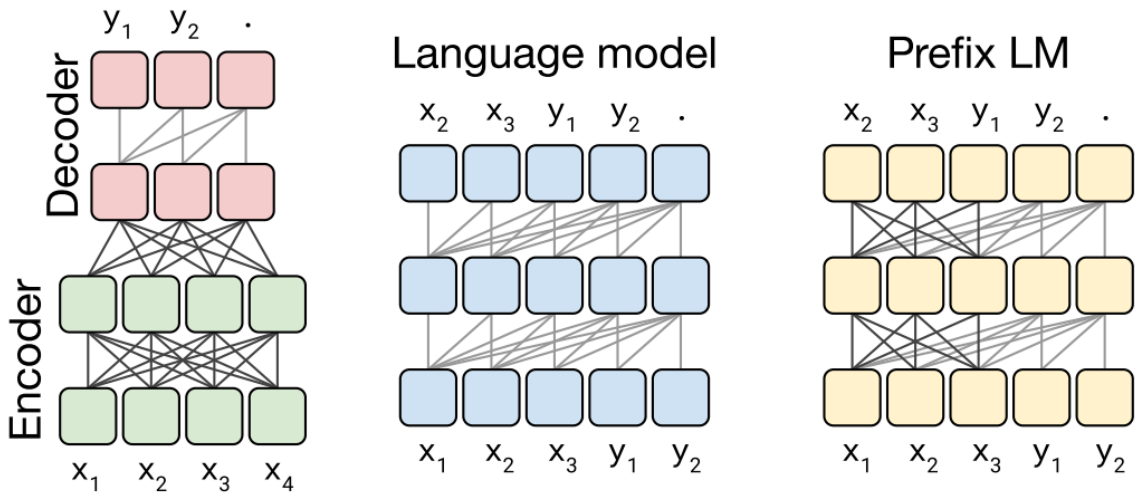
- Variants:
  - Encoder-decoder
  - Language model
  - Prefix LM: BERT, ULM





Model Architectures

- L + L Layer Encoder-decoder vs. L Layer Language model
  - 2x parameters
  - Same computation cost
- Ablation Study:
  - Share parameter across Encoder and Decoder
  - L/2 + L/2 Layer Encoder-decoder



Model Architectures: Results

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39
Encoder-decoder	LM	$2P$	M	79.56	18.59	76.02	64.29	26.27	39.17	26.86
Enc-dec, shared	LM	P	M	79.60	18.13	76.35	63.50	26.62	39.17	27.05
Enc-dec, 6 layers	LM	P	$M/2$	78.67	18.26	75.32	64.06	26.13	38.42	26.89
Language model	LM	P	M	73.78	17.54	53.81	56.51	25.23	34.31	25.38
Prefix LM	LM	P	M	79.68	17.84	76.87	64.86	26.28	37.51	26.76

- Surprisingly, sharing parameters across the encoder and decoder performed nearly as well. (ALBERT)
- and better than prefix LM. Explicit encoder-decoder structure is useful.
- Denoising objective > LM objective

Empirical Survey

Methodology “coordinate descent”

Baseline → Architecture → **Objective** → Dataset

→ Transfer Approach → Scaling

Unsupervised Objectives

Objective	Inputs	Targets
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style	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)*
I.i.d. noise, mask tokens	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>

Unsupervised Objectives

- LM vs. Masked LM vs. Deshuffling

Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
Prefix language modeling	80.69	18.94	77.99	65.27	**26.86**	39.73	**27.49**
BERT-style [Devlin et al., 2018]	**82.96**	**19.17**	**80.65**	**69.85**	**26.78**	**40.03**	**27.41**
Deshuffling	73.17	18.59	67.61	58.47	26.11	39.30	25.62

Table 4: Performance of the three disparate pre-training objectives described in Section 3.3.1.

Unsupervised Objectives

- Masked LM
  - BERT-style: 15% → (90% [MASK], 10% [Random Token])
  - MASS-style: 15% → [MASK]
  - replace spans — Thank you <X> me to your party <Y> week . — <X> for inviting <Y> last <Z>
  - drop tokens — Thank you me to your party week . — for inviting last

Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
BERT-style [Devlin et al., 2018]	82.96	19.17	**80.65**	69.85	26.78	**40.03**	27.41
MASS-style [Song et al., 2019]	82.32	19.16	80.10	69.28	26.79	**39.89**	27.55
★ Replace corrupted spans	83.28	**19.24**	**80.88**	**71.36**	**26.98**	39.82	**27.65**
Drop corrupted tokens	**84.44**	**19.31**	**80.52**	68.67	**27.07**	39.76	**27.82**

Short Target &
Fast Training

Due to CoLA

Unsupervised Objectives

- Corruption rate
  - Not Sensitive

Corruption rate	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
10%	**82.82**	19.00	**80.38**	69.55	**26.87**	39.28	**27.44**
★ 15%	**83.28**	19.24	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
25%	**83.00**	**19.54**	**80.96**	70.48	**27.04**	**39.83**	**27.47**
50%	81.27	19.32	79.80	70.33	**27.01**	**39.90**	**27.49**

Table 6: Performance of the i.i.d. corruption objective with different corruption rates.

Unsupervised Objectives

- i.i.d corruption vs. span corruption (SpanBERT)
  - Many small spans vs. Little large spans
  - Long target vs. Short target (No. of spans + No. of masked tokens)
  - Slow vs. Fast

Slightly but significantly improvement

Span length	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline (i.i.d.)	**83.28**	19.24	80.88	71.36	**26.98**	**39.82**	**27.65**
2	**83.54**	19.39	**82.09**	**72.20**	**26.76**	**39.99**	**27.63**
3	**83.49**	**19.62**	**81.84**	**72.53**	**26.86**	39.65	**27.62**
5	**83.40**	19.24	**82.05**	**72.23**	**26.88**	39.40	**27.53**
10	82.85	19.33	**81.84**	70.44	**26.79**	39.49	**27.69**

Table 7: Performance of the span-corruption objective (inspired by Joshi et al. [2019]) for different average span lengths. In all cases, we corrupt 15% of the original text sequence.

Unsupervised Objectives

- Message:
  - Small modification to the masked language model objective may not leads to significant improvement.
  - Try something different!

Empirical Survey

Methodology “coordinate descent”

Baseline → Architecture → Objective → **Dataset**

→ Transfer Approach → Scaling

Pre-training Datasets

- C4: Common Crawl with heuristic filterin
- Unfiltered C4: Common Crawl only use use *langdetect* to extract English text
- RealNews-like: omitted any non-news content in C4
- WebText-like (GPT2-like): high Reddit score webpages in C4
- Wikipedia
- Wikipedia + Toronto Books Corpus (BERT)

Pre-training Datasets

Pre-training on in-domain unlabeled data can improve performance on downstream tasks.

Dataset	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**

Table 8: Performance resulting from pre-training on different datasets. The first four variants are based on our new C4 dataset.

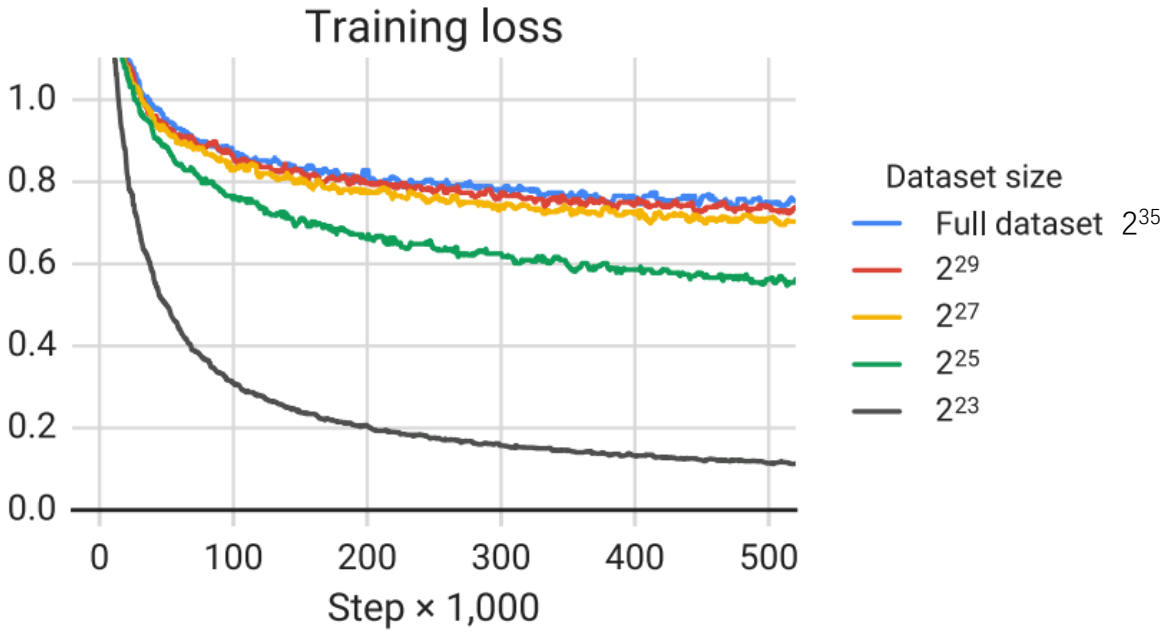
Due to ReCoRD,
News domain

Due to MultiRC,
the same domain as TBC

SQuAD, from Wikipedia

Pre-training Datasets

- Size
  - The larger the better



Number of tokens	Repeats	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Full dataset 2^{35}	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

Empirical Survey

Methodology “coordinate descent”

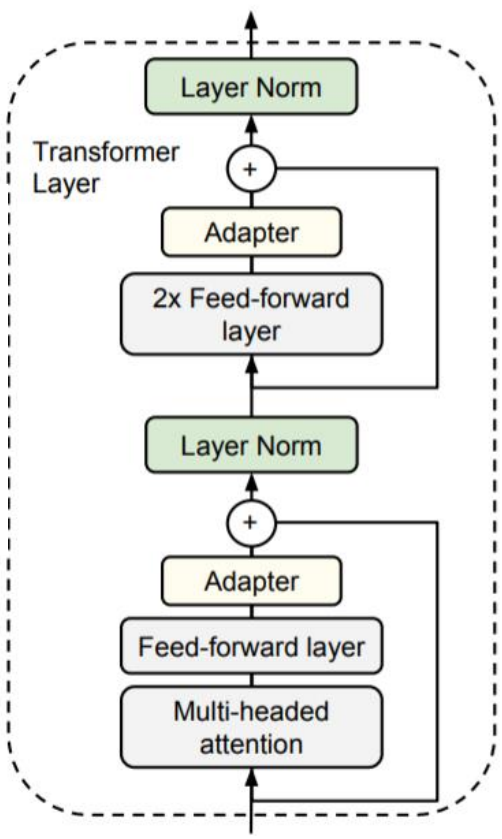
Baseline → Architecture → Objective → Dataset

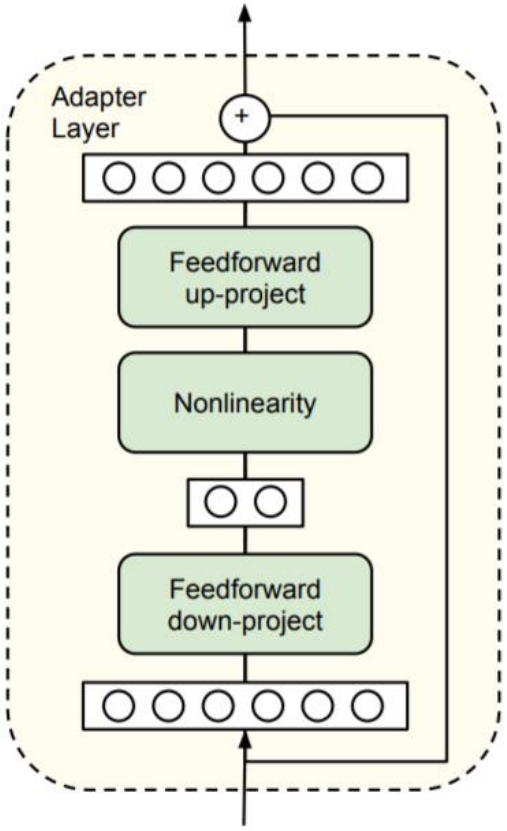
→ **Transfer Approach** → Scaling

Transfer Approaches

- Adaptive Layers (Houlsby 2019):
  - Only adaptive layers are updated

Fine-tuning method	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ All parameters	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Adapter layers, $d = 32$	80.52	15.08	79.32	60.40	13.84	17.88	15.54
Adapter layers, $d = 128$	81.51	16.62	79.47	63.03	19.83	27.50	22.63
Adapter layers, $d = 512$	81.54	17.78	79.18	64.30	23.45	33.98	25.81
Adapter layers, $d = 2048$	81.51	16.62	79.47	63.03	19.83	27.50	22.63
Gradual unfreezing	82.50	18.95	79.17	**70.79**	26.71	39.02	26.93





Transfer Approaches

- Gradual Unfreezing (ULMFiT):
  - First unfreeze the last layer → the next lower layer

Fine-tuning method	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ All parameters	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Adapter layers, $d = 32$	80.52	15.08	79.32	60.40	13.84	17.88	15.54
Adapter layers, $d = 128$	81.51	16.62	79.47	63.03	19.83	27.50	22.63
Adapter layers, $d = 512$	81.54	17.78	79.18	64.30	23.45	33.98	25.81
Adapter layers, $d = 2048$	81.51	16.62	79.47	63.03	19.83	27.50	22.63
Gradual unfreezing	82.50	18.95	79.17	**70.79**	26.71	39.02	26.93

Transfer Approaches

- Multi-task learning:
  - Examples-proportional mixing: $r_m \propto s_m$
  - Temperature-scaled mixing (Multilingual BERT): $r_m \propto s_m^{1/T}$
  - Equal mixing: $r_m \propto 1$ worst

Mixing strategy	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline (pre-train/fine-tine)	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Equal	76.13	19.02	76.51	63.37	23.89	34.31	26.78
Examples-proportional, $K = 2^{16}$	80.45	19.04	77.25	69.95	24.35	34.99	27.10
Examples-proportional, $K = 2^{17}$	81.56	19.12	77.00	67.91	24.36	35.00	27.25
Examples-proportional, $K = 2^{18}$	81.67	19.07	78.17	67.94	24.57	35.19	27.39
Examples-proportional, $K = 2^{19}$	81.42	**19.24**	79.78	67.30	25.21	36.30	**27.76**
Examples-proportional, $K = 2^{20}$	80.80	**19.24**	**80.36**	67.38	25.66	36.93	**27.68**
Examples-proportional, $K = 2^{21}$	79.83	18.79	79.50	65.10	25.82	37.22	27.13
Temperature-scaled, $T = 2$	81.90	**19.28**	79.42	69.92	25.42	36.72	27.20
Temperature-scaled, $T = 4$	80.56	**19.22**	77.99	69.54	25.04	35.82	27.45
Temperature-scaled, $T = 8$	77.21	19.10	77.14	66.07	24.55	35.35	27.17

Transfer Approaches

- Pretrain → Multi-task learning → Single-task fine tune (MTDNN)

Training strategy	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Unsupervised pre-training + fine-tuning	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	39.82	27.65
Multi-task training	81.42	**19.24**	79.78	67.30	25.21	36.30	27.76
Multi-task pre-training + fine-tuning	**83.11**	**19.12**	**80.26**	**71.03**	**27.08**	39.80	**28.07**
Leave-one-out multi-task training	81.98	19.05	79.97	**71.68**	**26.93**	39.79	**27.87**
Supervised multi-task pre-training	79.93	18.96	77.38	65.36	26.81	**40.13**	**28.04**

Empirical Survey

Methodology “coordinate descent”

Baseline → Architecture → Objective → Dataset

→ Transfer Approach → **Scaling**

Scaling

- With similar computation cost
  - increasing the training time and increasing the model size can be complementary

Scaling strategy	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
Baseline	83.28	19.24	80.88	71.36	26.98	39.82	27.65
1× size, 4× training steps	85.33	19.33	82.45	74.72	27.08	40.66	27.93
1× size, 4× batch size	84.60	19.42	82.52	74.64	27.07	40.60	27.84
2× size, 2× training steps	**86.18**	19.66	**84.18**	77.18	27.52	**41.03**	28.19
4× size, 1× training steps	**85.91**	19.73	**83.86**	**78.04**	27.47	40.71	28.10
4× ensembled	84.77	**20.10**	83.09	71.74	**28.05**	40.53	**28.57**
4× ensembled, fine-tune only	84.05	19.57	82.36	71.55	27.55	40.22	28.09

Table 13: Comparison of different methods of scaling up our baseline model. All methods except ensembling fine-tuned models use 4× the computation as the baseline. "Size" refers to the number of parameters in the model and "training time" refers to the number of steps used for both pre-training and fine-tuning.

State-of-the-Art

Baseline + Architecture + Objective + Dataset

+ Transfer Approach + Scaling

SOA model

- Objective: span-corruption (SpanBERT)
- Longer training: 1M steps + 2048 batch size → 1T tokens
  - 8x BERT, 2x XLNet, ½ x RoBERTa
- Model sizes:
  - Small: 60M Base: 220M Large: 770M XLarge: 3B XXLarge: 11B
- Multi-task pre-training: √
- Finetune on GLUE and SuperGLUE: 8 batch size

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	**89.7**	**70.8**	**97.1**	91.9	89.2	92.5	92.1

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	74.6	90.4	**92.0**	**91.7**	96.7	**92.5**	**93.2**

Model	SQuAD EM	SQuAD F1	SuperGLUE Average	BoolQ Accuracy	CB F1	CB Accuracy	COPA Accuracy
Previous best	88.95[d]	94.52[d]	84.6[e]	87.1[e]	90.5[e]	95.2[e]	90.6[e]
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	**90.06**	**95.64**	**88.9**	**91.0**	**93.0**	**96.4**	**94.8**

Model	MultiRC F1a	MultiRC EM	ReCoRD F1	ReCoRD Accuracy	RTE Accuracy	WiC Accuracy	WSC Accuracy
Previous best	84.4[e]	52.5[e]	90.6[e]	90.0[e]	88.2[e]	69.9[e]	89.0[e]
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	**88.2**	**62.3**	**93.3**	**92.5**	**92.5**	**76.1**	**93.8**

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**[f]	**43.8**[f]	**38.5**[g]	43.47[h]	20.30[h]	40.63[h]
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**

Table 14: Performance of our T5 variants on every task we study. Small, Base, Large, 3B, and 11B refer to model configurations with 60 million, 220 million, 770 million, 3 billion, and 11 billion parameters, respectively. In the first row of each table, we report the state-of-the-art for the task, with the superscript denoting its source with references listed at the end of this caption. All results are reported on the test set except for SQuAD where we use the validation set. [a][Lan et al., 2019] [b][Wang et al., 2019c] [c][Zhu et al., 2019] [d][Yang et al., 2019] [e][Liu et al., 2019c] [f][Edunov et al., 2018] [g][Lample and Conneau, 2019] [h][Dong et al., 2019]