Are Pre-trained Convolutions Better than Pre-trained Transformers?

Yi Tay
Google Research
Mountain View, California
yitay@google.com

Mostafa Dehghani
Google Research, Brain Team
Amsterdam, Netherlands
dehghani@google.com

Jai Gupta
Google Research
Mountain View, California
jaigupta@google.com

Vamsi Aribandi*
Google Research
Mountain View, California
aribandi@google.com

Dara Bahri
Google Research
Mountain View, California
dbahri@google.com

Zhen Qin
Google Research
Mountain View, California
zhenqin@google.com

Donald Metzler
Google Research
Mountain View, California
metzler@google.com

Abstract

In the era of pre-trained language models, Transformers are the de facto choice of model architectures. While recent research has shown promise in entirely convolutional, or CNN, architectures, they have not been explored using the pre-train-fine-tune paradigm. In the context of language models, are convolutional models competitive to Transformers when pre-trained? This paper investigates this research question and presents several interesting findings. Across an extensive set of experiments on 8 datasets/tasks, we find that CNN-based pre-trained models are competitive and outperform their Transformer counterpart in certain scenarios, albeit with caveats. Overall, the findings outlined in this paper suggest that conflating pre-training and architectural advances is misguided and that both advances should be considered independently. We believe our research paves the way for a healthy amount of optimism in alternative architectures.

1 Introduction

In the modern era of pre-training, there appears to be an unbreakable tie between Transformer architectures (Vaswani et al., 2017) and pre-trained language models. Models such as BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), and T5 (Raffel et al., 2019) have all adopted Transformers as their underlying architecture. As a matter of fact, there are barely any recent pre-trained models not based on Transformers.

While the contextual representation learning has a rich history (Pennington et al., 2014; Dai and Le, 2015; Chidambaram et al., 2018; Liu et al., 2020; Qiu et al., 2020), modern pre-trained language modeling started with models like ELMo (Peters et al., 2018) and CoVE (McCann et al., 2017) which are based on recurrent (e.g. LSTM (Hochreiter and Schmidhuber, 1997)) architectures. Although they were successful, research using these architectures dwindled as Transformers stole the hearts of the NLP community, having, possibly implicitly, been perceived as a unequivocal advancement over its predecessors.

Recent work demonstrates the promise of entirely convolution-based models (Wu et al., 2019; Gehring et al., 2017) and questions the necessity of self-attentive architectures like Transformers. For example, in (Wu et al., 2019), the proposed convolutional seq2seq models outperform Transformers on a series of canonical benchmarks such as machine translation and language modeling. From these findings emerge a rather natural line of questioning - should we consider pre-trained models beyond Transformers?

Despite early success, the relevance of convolutional models in the era of pre-trained language models remains an open question. To the best of our knowledge, convolutional architectures have not yet been rigorously evaluated under the *pre-train-fine-tune* paradigm. This is the primary purpose of this work. Concretely, this paper seeks to empirically validate whether pre-trained convolutions are competitive with pre-trained Transformers across a range of tasks.

The interaction between pre-training schemes and model architectures is an under-studied topic. Are only Transformers able to capitalize on the

*Google AI Resident

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 4349–4359
August 1–6, 2021.

benefits of pre-training? If we use a different architectural inductive bias, would there also be a substantial gain unlocked by pre-training? Are pre-trained convolutions better in particular scenarios? This paper investigates these questions.

There are a number of obvious benefits of convolution-based models. Firstly, convolutions do not suffer from the quadratic memory complexity of self-attention - a problem significant enough that it spawned the creation of the entirely new category of "efficient" Transformer architectures (Tay et al., 2020b, 2021). Secondly, convolutions operate locally and do not rely on positional encodings as an order signal to the model. That said, convolutions also come with a slew of downsides. For example, being unable to access global information means such models are unable to perform a form of cross-attention across multiple sequences. We dive into the details of this more in subsequent sections.

In this paper, we present a pre-trained convolutional sequence-to-sequence, or Seq2Seq, model. We train our convolutional model using span-based sequence-to-sequence denoising objectives similar to those employed in T5 (Raffel et al., 2019). We evaluate a variety of convolutional variants (e.g., dilated, lightweight, dynamic (Wu et al., 2019), etc.) under both raw (no pre-training) and pre-train-fine-tune paradigms. Our goal is to understand the true competitiveness of convolutional architectures in the era of pre-training.

We show that pre-trained convolutions are competitive against pre-trained Transformers via a set of experiments on a potpourri of NLP tasks, like toxicity detection, sentiment classification, news classification, query understanding and semantic parsing/compositional generalization (Kim and Linzen, 2020). Moreover, we find that pre-trained convolutions can outperform, in terms of model quality and training speed, state-of-the-art pre-trained Transformers (Raffel et al., 2019) in certain scenarios. However, to provide a balanced perspective, we also describe scenarios where pre-trained convolutions do not perform well and may be deemed unsuitable.

Contributions Overall, the main contributions of this paper can be summarized as follows:

- We perform a comprehensive empirical evaluation of convolutional Seq2Seq models under the *pre-train-fine-tune* paradigm. To the best of our knowledge, the competitiveness and relevance of pre-trained convolutions still remains an open question.
- We make several important observations. Specifically, we find that (1) pre-training helps convolutional models just as much as it helps Transformers, and (2) pre-trained convolutions are competitive alternatives in certain scenarios in terms of model quality and training speed.
- We conduct extensive experiments across 8 datasets spanning a diverse range of tasks and domains. On 7 out of 8 tasks, we find that pre-trained convolutions outperform a recent state-of-the-art transformer (T5 (Raffel et al., 2019)) with and without pre-training. We examine the speed and operation count (FLOPS) of convolutions versus Transformers and find that convolutions are not only faster but also scale better to longer sequence lengths.

2 Related Work

Pre-training on a large corpus has become the primary method of learning universal language representations to solve different downstream NLP tasks. The first generation of pre-trained models aimed at learning embedding for words, like Skip-Gram (Mikolov et al., 2013) and Glove (Pennington et al., 2014), and quickly developed to learning contextualized representation for words, like ELMO (Peters et al., 2018), GPT (Radford et al., 2018), and BERT (Devlin et al., 2018). This, however, is not the only axis in which pre-trained models have evolved.

Different objective functions and various tasks, both supervised and unsupervised, have been explored for pre-training. For instance, CoVe (McCann et al., 2017) uses machine translation as the pre-training task, ELMO (Peters et al., 2018) and GPT (Radford et al., 2018) use language modeling objectives, BERT (Devlin et al., 2018) uses masked language modeling, T5 (Raffel et al., 2019) and MASS (Song et al., 2019) use Seq2Seq masked language modeling, and XLNet (Yang et al., 2019) utilizes permuted language modeling. In addition to this, BART (Lewis et al., 2019) uses a denoising autoencoder setup during pre-training, where the model takes a partially corrupted input and is trained to recover the original, undistorted input. Some models use a contrastive learning setup during pertaining, like replaced token detection, used

by ELECTRA (Clark et al., 2020), and sentence order prediction, used by ALBERT (Lan et al., 2019) and StructBERT (Wang et al., 2019).

Another axis where pre-trained models in NLP explored different ideas is model architecture. ELMO (Peters et al., 2018) and CoVe (McCann et al., 2017) used LSTMs as the base model. Later, Transformers (Vaswani et al., 2017) became the de facto architecture of pre-trained NLP models. BERT (Devlin et al., 2018), XLNet (Yang et al., 2019) and RoBERTa (Liu et al., 2019) use the Transformer encoder, while GPT (Radford et al., 2018), GPT-2 (Radford et al.), and GPT-3 (Brown et al., 2020) use the Transformer decoder as the backbone. Some pre-trained models are also are based on the encoder-decoder transformer architecture, like T5 (Raffel et al., 2019), MASS (Song et al., 2019), and BART (Lewis et al., 2019). In this paper, we investigate another model architecture variation by studying the power of convolutional neural network as the backbone of pre-trained models for NLP.

Convolutions have always been an interesting choice for sequence modeling and NLP applications (Kim, 2014; Bai et al., 2018; Kalchbrenner et al., 2016). Convolutions are lightweight and fast and have many interesting use-cases, notably for lightweight classification. In the era when LSTMs were the workhorses of NLP applications, convolutions were positioned nicely on the pareto frontier of the compute-performance curve. They are fast and lightweight, and unlike Transformers, they do not suffer from quadratic complexity. Our work is also well-aligned with the resurgence of interest in convolutions where (Wu et al., 2019) showed that convolutions can outperform self-attention on several sequence transduction tasks. Moreover, the necessity of the self-attention inductive bias in transformers have been also a subject of recent interest. Synthesizer models (Tay et al., 2020a) showed that transformers can still do pretty well without token-token dot product self-attention and a random attention matrix can perform competitively on certain tasks.

3 Pre-Trained Convolution Models

This section describes the pre-trained Convolution Model. For most of our experiments, we adopt depthwise separable convolutions (Kaiser et al., 2017; Sifre and Mallat, 2014; Chollet, 2017) which have shown to be fast and efficient variants of the standard convolution.

3.1 Lightweight Depthwise Convolution

This section introduces Lightweight Depthwise Convolutions (Wu et al., 2019) which forms the backbone of our pre-trained convolution model.

3.1.1 Depthwise convolutions

Depthwise convolutions convolve independently over every channel. Given an input tensor X of dimensions $n \times d$, the depthwise convolution, $D(X, W_{c,:}, i, c)$ is defined as:

$$O_{i,c} = \sum_{j-1}^{k} W_{c,j} \cdot X_{i+j-\lceil \frac{k+1}{2} \rceil), c} \quad (1)$$

where $W \in \mathbb{R}^{d \times k}$ are the learnable parameters of the layer. $O_{i,c}$ is the output at position i and channel c. The overall output is a tensor of $n \times d$ of identical shape as the input.

3.1.2 Lightweight Convolutions

$L(.)$ are depthwise separable convolutions with (1) softmax-normalized kernels and (2) shared output channels and weight tying. Specifically, this is written as:

$$O^{L}_{i,c} = \sum_{j-1}^{k} \text{softmax}(W_{\hat{c},j}) \cdot X_{i+j-\lceil \frac{k+1}{2} \rceil), \hat{c}} \quad (2)$$

where $\hat{c} = \frac{cH}{d}$. In short, parameters are shared every $\frac{d}{H}$ output channels. When $H = 1$, this is equivalent to sharing all the weights of all channels.

3.1.3 Dynamic Convolutions

Dynamic Convolutions $D_Y(.)$ are a new form of lightweight convolutions introduced by (Wu et al., 2019). The key idea is to learn position-specific kernels for performing lightweight convolutions. This can be written as:

$$D_Y = L(X, f(X_i)_{h,:}, i, c), \quad (3)$$

where $f(.)$ is a linear transformation with parameters $W^Q \in \mathbb{R}^{H \times k \times d}$ that learns a position dependent kernel.

3.2 Span-based Seq2Seq pre-training

We adopt span-based sequence-to-sequence pre-training as per (Raffel et al., 2019). Specifically, given an input sequence, we randomly mask spans of lengths L and replace them with a special sentinel token. The pre-training task is then to generate the masked tokens as targets. For example: *Inputs: The happy cat sat [mask].* and *Outputs: on the mat.*

3.2.1 Convolutional Seq2Seq Architecture

We implement a Seq2Seq (Sutskever et al., 2014) architecture similar to (Wu et al., 2019). The key difference when compared with Transformer architectures is that we replace the multi-headed self-attention with convolutional blocks. Instead of query-key-value transforms, we use gated linear unit projections following (Wu et al., 2019). Each convolution block be written as:

$$X^1 = W^I X \odot \text{sigmoid}(W^S X),$$
$$X^2 = \text{ConvBlock}(X^1),$$
$$X^3 = W^O(X^2),$$

where W^I, W^S, W^O are trainable parameters. We experiment with simple lightweight convolutions, dynamic convolutions and dilated convolutions in our experiments. Following (Wu et al., 2019; Gehring et al., 2017), the encoder-decoder attention remains untouched. The convention follows the backbone Transformer model in which we wrap each submodule with layer normalization and residual connectors. Hence, each Conv block is written as:

$$X_A = \text{LayerNorm}(\text{Conv}(X)) + X,$$
$$X_B = \text{LayerNorm}(\text{FFN}(X_A) + X_A,$$

where Conv is any of the convolution models that we explore in our experiments. FFN(.) is a two layer feed-forward network with ReLU activations in the middle.

3.2.2 Optimization

The model optimizes the token-wise cross-entropy loss and is trained with teacher forcing.

$$L = \sum_{t=1}^{L} \sum_{i=1}^{n} \log(\pi_i^t) + (1 - y_i^t) \log(1 - \pi_i^t),$$

where π_i^t is the prediction of class i at time step t and y_i^t is the ground truth label of the class i at time step t.

4 Research Questions and Discussion

Before we delve into our experiments, we establish a set of research questions and agenda we hope this work aims to bring clarity to.

- **RQ1**: Do convolutions benefit from pre-training as much as Transformers?
- **RQ2**: Are convolutional models, pre-trained or otherwise, competitive with Transformer models? When do they perform well?
- **RQ3**: What are the benefits (if any) of using pre-trained convolution models over pre-trained Transformers? Are convolutions faster alternatives to self-attention based Transformers?
- **RQ4**: What are the failure modes, caveats and reasons to **not** use pre-trained convolutions?
- **RQ5**: Are certain convolution variants better than others?

5 Experiments and Analysis

This section presents our analysis and results.

5.1 Datasets

Our evaluation is based on the following datasets and tasks.

- **Toxicity Detection** - We use the CIVIL COMMENTS (Borkan et al., 2019) and WIKI TOXIC SUBTYPES dataset (Wulczyn et al., 2017). Given a piece of short text (originating from social media or wikipedia), the goal is to determine if the content is toxic, i.e., a binary classification task. For this task, we evaluate on both accuracy and F1 score.
- **Sentiment Classification** - This is a binary classification task that determines the polarity of documents, sentences and/or tweets. We use the IMDb reviews dataset (Maas et al., 2011), Stanford Sentiment Treebank (SST-2) (Socher et al., 2013) dataset, along with Twitter Sentiment140 (S140) (Go et al., 2009) dataset.
- **News Classification** - This is a task of topic categorization for news articles. We use the AGNews dataset (Zhang et al., 2015). This is a four-way classification task.
- **Question Classification** We use the TREC fine-grained question classification dataset (Li and Roth, 2002). This task involves classifying questions into 46 fine-grained question categories.
- **Semantic Parsing / Compositional Generalization** Compositional generalization is the

ability of models to generalize *compositionally* outside of the training distribution. To be specific, it needs be able to handle unseen combinations at test time. For this task, we use the COGS dataset (Kim and Linzen, 2020), a task of generating semantic representation of a given English sentence. For example, *A cat smiled* $\rightarrow$ cat(x_1) AND smile.agent(x_2, x_1).

All of the datasets, with the exception of the recent COGS dataset (Kim and Linzen, 2020), are Tensorflow datasets[1].

For each dataset, we evaluate all models with and without pre-training (details in subsequent sections). Table 1 reports the statistics of the datasets used in this paper.

Dataset / Task	# Train	# Test	# Class
Civil Comments	3,820,210	205,781	2
Wiki Toxicity	561,808	234,564	2
IMDb	25,000	25,000	2
SST-2	67,000	1,800	2
S140	1,600,000	359	2
TREC	4,500	500	46
AGNews	120,000	7,600	4
COGS	24,000	3000	N/A

Table 1: Statistics of datasets used in our experiments. Datasets are diverse in terms of domains, tasks and amount of labeled data.

5.2 Experimental Setup

This section describes our experimental setup.

5.2.1 Models

Our models are largely based on sequence to sequence models, a paradigm that has demonstrated great success made evident by models such as BART (Lewis et al., 2019) and T5(Raffel et al., 2019). We implement our models in Mesh Tensorflow (MTF) (Shazeer et al., 2018), a library for distributed and efficient parallel model training that has similar API to Tensorflow. We train models that are of base size, which corresponds to 12 layers each in the encoder and decoder, along with 3072 dimensions for the feed-forward layers, a model dimension of 768 and a total of 12 heads. Our Transformer models are largely based on T5 (Raffel et al., 2019), which is considered the current state-of-the-art Transformer model for NLP tasks and hence serves as a strong baseline. For the convolution models, our lightweight convolution and dynamic convolution models have a window size[2] of 7 across all layers, the number of unique depth filters is 2. For dilated models, we use a filter size of $[4, 4, 7, 7, 15, 15, 15, 15, 31, 31, 31]$ for our 12 layer convolution model.

5.2.2 Pre-training

We pre-train both our convolutional and Transformer models for 524K steps with a batch size of 128. Given the input sequence length of 512, this corresponds to 65536 tokens per batch. For pre-training, we use the Colossal Cleaned CommonCrawl Corpus (C4) (Raffel et al., 2019) dataset which has demonstrated impressive results on downstream tasks. We use the span based seq2seq objective as the pre-training objective as mentioned in earlier sections. The span size is set to 3 and a corruption rate of 15% is adopted. We use the Adafactor optimizer (Shazeer and Stern, 2018) with an inverse square root learning rate scheduler. Each pre-training run is performed using 16 TPU-v3 chips and takes approximately 12 hours to complete for models of base size.

5.2.3 Downstream Fine-tuning

We fine-tune the pre-trained models using the following set of hyperparameters: We use a constant learning rate which is tuned amongst $\{0.001, 0.0005, 0.0001\}$. The batch size is generally set to 64 but occasionally set to 32 for smaller datasets. Intuitively, sequence length is task dependent but generally approximately the 90th percentile for each task. We fine-tune for a maximum of $100K$ steps and report peak validation performance. Fine-tuning uses the same Adafactor optimizer as during training. We perform fine-tuning on similar hardware, i.e., typically 16 TPUv3 chips are used per fine-tuning job.

5.3 Experimental Results

This section describes our experimental setup and results.

5.4 Results on Toxicity Detection

Table 2 reports results on toxicity detection. On both toxicity detection datasets the pre-trained and no-pre-training (raw) setup, the best models are the dilated convolution models and the dynamic convolution models. In fact, all convolutional models

[1] https://www.tensorflow.org/datasets/catalog/overview.

[2] We believe that tuning the hyperparameters of the convolution models can result in even better performance. However, we decided to keep these hyperparameters simple for the start.

outperform Transformers on both CivilComments and WikiToxic. Before pre-training, convolutions outperform Transformers by approximately 1.5 absolute percentage points. The gap narrows after pre-training where Transformers see a better gain (e.g., $+5.1\%$ against $+4.3\%$) from pre-training over convolutions on the CivilComments dataset. However, the converse is true on WikiToxic - the only case of performance degradation after pre-training. Overall, on this task, convolutions are competitive to Transformers and outperform them.

5.5 Results on Sentiment Classification

Results on Sentiment Classification (IMDb, SST-2 and S140) can be found in Table 2. On the IMDb reviews dataset, the best non-pre-trained model is the lightweight convolution model, outperforming the Transformer model. The best pre-trained model is the Transformer model. However, all convolutional models come in close with less than a percentage point gap difference with pre-trained Transformers. On the SST-2 and S140 tasks, we observe that the best models are convolution-based, regardless of whether the model is pre-trained or not.

5.6 Results on Question Classification

The best non-pre-trained model is the Lightweight Convolution model. For pre-trained models, convolutional models also outperform the pre-trained Transformer. On this task, while most models benefit significantly from pre-training, Transformers seem to benefit slightly more from pre-training.

5.7 Results on News Classification

Results on news classification seems to follow similar trends as other benchmarks. Convolutional models outperform Transformers both in non-pre-trained and pre-trained setups. The highest gain from pre-training is obtained from the dilated convolution model.

5.8 Results on Compositional Generalization Challenge and Semantic Parsing

We conduct additional experiments on semantic parsing and compositional generalization. The task is framed as a sequence generation task. We use the recently proposed (Kim and Linzen, 2020) dataset. On the in-distribution test set, Transformers and convolutions have identical performance (95%). On the generalization or out of distribution set, Transformers perform at 77.5% while convolutions come in at 76.9. While convolutions do not exactly outperform Transformers, they come in close enough to be considered competitive.

5.9 Summary of Results

On the seven tasks across a broad range of domains we find that (1) non-pre-trained convolutions are competitive and frequently outperform non-pre-trained Transformers, (2) pre-trained convolutions outperform pre-trained Transformers on six out of seven tasks. This answers **RQ2**.

We also find that convolutions are able to benefit from pre-training, in a similar fashion to self-attention-based models. Hence, the benefits achieved by pre-training are not exclusive to Transformer models. This answers **RQ1**.

Amongst the pre-trained convolutional models, we find that dilated convolutions and dynamic convolutions are generally better than lightweight convolutions, thus answering **RQ5**.

Finally, we observe that relative performance (i.e., rankings) do change with pre-training. This definitely shows that there is some kind of effect from composing architectures with pre-training. The direct implication of this effect is that a model that performs well (relatively) without pre-training will not necessarily perform the best when pre-trained (and vice versa). Hence, aside from conflating architectures with pre-training schemes, we do also need to take note that different architectures may behave differently under pre-training.

6 Discussion and Analysis

This section expands on the results via a detailed analysis and discussion. We discuss the pros/cons of pretrained convolutions, the impact of pre-training on performance and also recommendations to the broader community.

6.1 When do we expect pre-trained convolutions to fail?

In our experimental section, we observed the potential upsides of convolutional models over well-established pre-trained Transformers and observe that we are able to get quality improvements in certain cases. However, it might be good to further understand the drawbacks of convolutions.

One obvious weakness of pre-trained convolutions are their **lack of cross-attention** inductive bias that comes for *free* with self-attention in the Transformer encoder. For this reason, it is not a

Model	CIVILCOMMENT Acc	CIVILCOMMENT F1	WIKITOXIC Acc	WIKITOXIC F1	IMDb Acc	SST-2 Acc	S140 Acc	TREC Acc	News Acc
No pre-training									
Trans.	77.22	85.09	91.93	95.45	84.81	78.44	58.84	78.00	84.25
Light	78.58	85.82	91.05	94.65	**85.88**	81.65	60.64	**82.20**	**87.22**
Dilat.	**79.94**	**86.50**	**92.29**	94.91	85.84	79.01	55.62	79.60	81.24
Dyna.	78.49	84.71	90.06	**95.66**	85.69	**82.80**	**60.84**	80.20	85.13
With pre-training									
Trans.	81.16	86.56	91.46	95.12	**94.16**	92.09	61.65	93.60	93.54
Light	81.47	87.58	93.61	96.48	93.60	**92.20**	61.65	93.60	93.63
Dilat.	81.67	**87.78**	**93.84**	96.21	93.92	92.09	**62.85**	**94.20**	93.26
Dyna.	**81.83**	87.71	93.76	**96.53**	93.35	91.59	62.45	92.40	**93.93**
Gain from pre-training									
Trans.	+5.1%	+1.7%	-0.6%	-0.4%	+11.0%	+17.4%	+4.7%	+20.0%	+11.0%
Light	+3.7%	+2.1%	+2.8%	+1.9%	+9.0%	+13.0%	+1.7%	+14.0%	+7.3%
Dilat.	+2.1%	+1.5%	+1.7%	+1.4%	+9.4%	+17.0%	+13.0%	+18.0%	+14.8%
Dyn.	+4.3%	+3.5%	+4.1%	+1.0%	+8.9%	+10.6%	+2.6%	+15.2%	+10.4%

Table 2: Comparison of pre-trained Convolutions and pre-trained Transformers on toxicity detection, sentiment classification, question classification and news classification. All models have approximately $230M$ parameters and are 12 layered seq2seq architectures. Our findings show that convolutions (1) also benefit from pretraining and (2) are consistently competitive to transformer models with and without pretraining.

good idea to use pre-trained convolutions for tasks that requires modeling the relationship between two or more sequences. To verify this, we run experiments on SQuAD and MultiNLI and find that convolutions do not come close to Transformers just because of this missing inductive bias. This should be clearly distinguished when examining and evaluating models, as how the early SNLI leaderboard[3] distinguished between models that used cross-attention and models that did not.

Our initial evaluations on benchmarks like SQuAD/MNLI (Rajpurkar et al., 2016; Williams et al., 2017) showed that pre-trained convolutions are indeed significantly lackluster. For example, convolutions only achieve $\approx 75\%$ accuracy on MultiNLI, while transformers easily achieve $\approx 84\%$ accuracy. Likewise, while transformers achieve about $\approx 90\%$ F1 on SQuAd, convolutions come in around $\approx 70\%$. This is entirely expected because there is no way the premise/question can interact with the hypothesis/context. (**RQ4**). However, our experiments show that this was only because they lack this cross-attention property. When we augment convolutions with a single layer of cross attention at the encoder, we find that pre-trained convolutions come close (a delta of $(\approx 1\%)$) to pre-trained Transformers on datasets such as MultiNLI (Williams et al., 2017), achieving about $\approx 83\%$ accuracy.

That said, we leave it to the practitioner to decide whether the cross-attention inductive bias is actually important for the problem at hand. We also like to emphasize that the pattern of concatenating sentence pairs is not necessary practical when scaling up since this requires inference on every permutation of sentence pairs. For this reason, dual encoder setups that do fast embedding space look-ups are more practical and feasible in practice (Guo et al., 2020). Given the strong performance of convolutions in a series of encoding tasks, we can expect pre-trained convolutions to do well in a dual encoder setup.

6.2 What are the benefits of pre-trained convolutions over Transformers?

We observed a reasonable quality improvement from using convolutions over Transformers. This section discusses the additional benefit.

6.2.1 Convolutions are faster and scale better to long sequences

Figure 1 reports training speed of convolution (LightConvs) versus transformers on a sequence to sequence task. The input lengths are varied from $\{64, 128, 256, 512, 1024, 2048, 4096\}$. We

[3] https://nlp.stanford.edu/projects/snli/

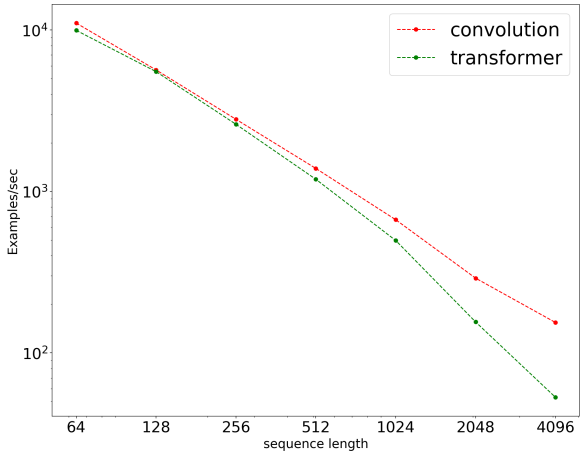

Figure 1: Effect of sequence length on processing speed (examples per second) on a seq2seq masked language modeling task. Results are benchmarked on 16 TPUv3 chips on C4 pre-training. Results are in log scale.

show that convolutions are not only consistently faster (even at shorter sequences) but scale better than transformers. Convolution scales linearly while transformers are not able to scale to longer sequences.

6.2.2 Convolutions are FLOPs efficient

We measure the number of FLOPs of convolutions versus transformers as we increase the sequence length. Figure 2 shows the phenomenon while varying sequence length. In general, across all sequence lengths, convolutions are more efficient in the number of floating point operations.

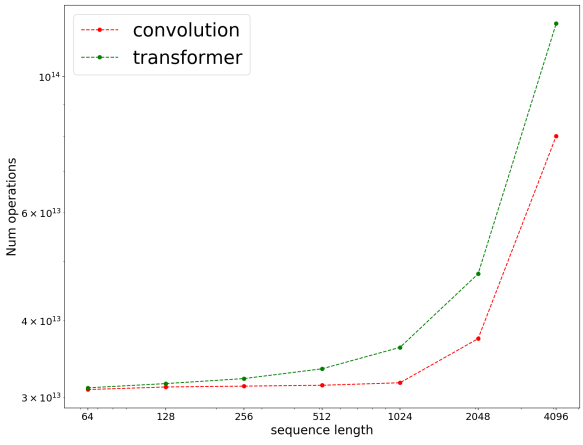

Figure 2: Effect of sequence length on number of FLOPs (einsum ops) on a seq2seq masked language modeling task. Results are benchmarked on 16 TPUv3 chips on C4 pre-training. Results are in log scale.

The overall findings that convolutions are faster both in wall clock time and in FLOPs answers **RQ3**. Moreover, we find that the FLOP efficiency of convolutions scales better across sequence lengths.

6.3 Are we suggesting to completely replace Transformers with convolution?

While Transformers have dominated the research landscape in NLP, this paper suggests that there are commonly overlooked benefits to convolutions such as model quality, speed, FLOPs and scalability. Moreover, it is previously unknown to whether convolutions benefit from pre-training. In this paper, we showed that they are competitive on **some** tasks and also benefit from pre-training in similar fashion to transformer models. However, on the flip side, we also highlighted that they are **unable** to handle tasks that require cross-attention or when there is a need to model > 1 sentence or documents within the same sequence. We believe that practitioners have good options and it might be worthwhile to explore architectures outside the well-established transformer models.

6.4 On not conflating pre-training with architectural advances

In this paper, we showed that three other (convolutional-based) architectures (e.g., lightweight, dymamic and dilated) also benefit from pre-training to the same extent as transformer models.

In the current research landscape, pre-training has always be tightly coupled and associated with transformers architectures. As a result, the success of BERT, transformers and large language models seem to be pretty conflated. While it is true that, to this date, the only model that large-scale pre-training has been applied to are transformer models, we believe there might be potential in other architectures.

Based on our empirical findings, we believe there is still significant room for the improving the understanding of the compositional effects of architecture and pre-training. Hence, we believe that the impact of this work extends beyond showing the competitiveness of convolution models in NLP. More concretely, the take home message is that there should be a healthy level of optimism in exploring architectural alternatives.

7 Conclusion

In this paper, we conducted an extensive study of the viability and feasibility of pre-trained convolu-

tions. Our experimental results show that convolutions can outperform Transformers in **both** pre-train and non-pre-trained setups. Our extensive experiments across 8 datasets spanning a diverse range of tasks, show that convolutions are able to benefit from pre-training to the same (or sometimes greater) extent than Transformers. While pre-trained transformers are the de-facto choice of architecture, our results show that they might not be the best in certain scenarios. Additionally, we discussed the caveats, trade-offs pertaining with runtime, scalability, number of FLOPS and model quality. Finally, we discussed the situations or data types that convolutions are not well equipped to handle and make an empirically informed recommendation for practitioners.

References

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019. Nuanced metrics for measuring unintended bias with real data for text classification. *CoRR*, abs/1903.04561.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Muthuraman Chidambaram, Yinfei Yang, Daniel Cer, Steve Yuan, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Learning cross-lingual sentence representations via a multi-task dual-encoder model. *arXiv preprint arXiv:1810.12836*.

François Chollet. 2017. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. *arXiv preprint arXiv:1511.01432*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12):2009.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Lukasz Kaiser, Aidan N Gomez, and Francois Chollet. 2017. Depthwise separable convolutions for neural machine translation. *arXiv preprint arXiv:1706.03059*.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*.

Najoung Kim and Tal Linzen. 2020. Cogs: A compositional generalization challenge based on semantic interpretation. *arXiv preprint arXiv:2010.05465*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Qi Liu, Matt J Kusner, and Phil Blunsom. 2020. A survey on contextual embeddings. *arXiv preprint arXiv:2003.07278*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in neural information processing systems*, pages 6294–6305.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, pages 1–26.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, et al. 2018. Mesh-tensorflow: Deep learning for supercomputers. In *Advances in Neural Information Processing Systems*, pages 10414–10423.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. *arXiv preprint arXiv:1804.04235*.

Laurent Sifre and Stéphane Mallat. 2014. Rigid-motion scattering for image classification.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. 2020a. Synthesizer: Rethinking self-attention in transformer models. *arXiv preprint arXiv:2005.00743*.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2021. Long range arena : A benchmark for efficient transformers. In *International Conference on Learning Representations*.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020b. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Jiangnan Xia, Liwei Peng, and Luo Si. 2019. Structbert: Incorporating language structures into pre-training for deep language understanding. *arXiv preprint arXiv:1908.04577*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 1391–1399, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification.