Pre-Training Transformers as Energy-Based Cloze Models

Kevin Clark[1] **Minh-Thang Luong**[2] **Quoc V. Le**[2] **Christopher D. Manning**[1]
[1]Stanford University [2]Google Brain
kevclark@cs.stanford.edu, thangluong@google.com
qvl@google.com, manning@cs.stanford.edu

Abstract

We introduce Electric, an energy-based cloze model for representation learning over text. Like BERT, it is a conditional generative model of tokens given their contexts. However, Electric does not use masking or output a full distribution over tokens that could occur in a context. Instead, it assigns a scalar energy score to each input token indicating how likely it is given its context. We train Electric using an algorithm based on noise-contrastive estimation and elucidate how this learning objective is closely related to the recently proposed ELECTRA pre-training method. Electric performs well when transferred to downstream tasks and is particularly effective at producing likelihood scores for text: it reranks speech recognition n-best lists better than language models and much faster than masked language models. Furthermore, it offers a clearer and more principled view of what ELECTRA learns during pre-training.

1 Introduction

The cloze task (Taylor, 1953) of predicting the identity of a token given its surrounding context has proven highly effective for representation learning over text. BERT (Devlin et al., 2019) implements the cloze task by replacing input tokens with [MASK], but this approach incurs drawbacks in efficiency (only 15% of tokens are masked out at a time) and introduces a pre-train/fine-tune mismatch where BERT sees [MASK] tokens in training but not in fine-tuning. ELECTRA (Clark et al., 2020) uses a different pre-training task that alleviates these disadvantages. Instead of masking tokens, ELECTRA replaces some input tokens with fakes sampled from a small generator network. The pre-training task is then to distinguish the original vs. replaced tokens. While on the surface it appears quite different from BERT, in this paper we elucidate a close connection between ELECTRA and cloze modeling. In particular, we develop a new way of implementing the cloze task using an energy-based model (EBM). Then we show the resulting model, which we call Electric, is closely related to ELECTRA, as well as being useful in its own right for some applications.[1]

EBMs learn an energy function that assigns low energy values to inputs in the data distribution and high energy values to other inputs. They are flexible because they do not have to compute normalized probabilities. For example, Electric does not use masking or an output softmax, instead producing a scalar energy score for each token where a low energy indicates the token is likely given its context. Unlike with BERT, these likelihood scores can be computed simultaneously for all input tokens rather than only for a small masked-out subset. We propose a training algorithm for Electric that efficiently approximates a loss based on noise-contrastive estimation (Gutmann and Hyvärinen, 2010). Then we show that this training algorithm is closely related to ELECTRA; in fact, ELECTRA can be viewed as a variant of Electric using negative sampling instead of noise-contrastive estimation.

We evaluate Electric on GLUE (Wang et al., 2019) and SQuAD (Rajpurkar et al., 2016), where Electric substantially outperforms BERT but slightly under-performs ELECTRA. However, Electric is particularly useful in its ability to efficiently produce pseudo-likelihood scores (Salazar et al., 2020) for text: Electric is better at re-ranking the outputs of a speech recognition system than GPT-2 (Radford et al., 2019) and is much faster at re-ranking than BERT because it scores all input tokens simultaneously rather than having to be run multiple times with different tokens masked out. In total, investigating Electric leads to a more principled understanding of ELECTRA and our results

[1] Code is available at https://github.com/google-research/electra

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 285–294,
November 16–20, 2020. ©2020 Association for Computational Linguistics

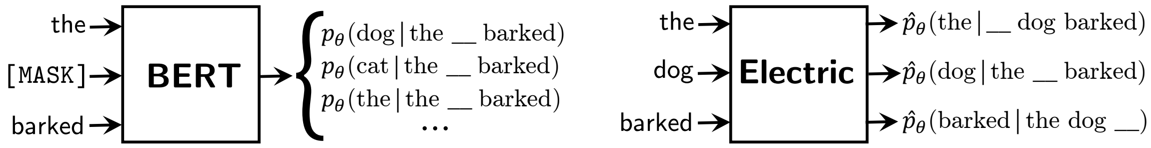
the
[MASK]
barked
BERT
p_θ(dog | the __ barked)
p_θ(cat | the __ barked)
p_θ(the | the __ barked)
...
the
dog
barked
Electric
$\hat{p}_\theta$(the | __ dog barked)
$\hat{p}_\theta$(dog | the __ barked)
$\hat{p}_\theta$(barked | the dog __)

Figure 1: Comparison of BERT and Electric. Both model the probability of a token given its surrounding context, but BERT produces a full output distribution over tokens only for masked positions while Electric produces un-normalized probabilities (but no full distribution) for all input tokens.

suggest that EBMs are a promising alternative to the standard generative models currently used for language representation learning.

2 Method

BERT and related pre-training methods (Baevski et al., 2019; Liu et al., 2019; Lan et al., 2020) train a large neural network to perform the cloze task. These models learn the probability $p_{\text{data}}(x_t|\boldsymbol{x}_{\backslash t})$ of a token x_t occurring in the surrounding context $\boldsymbol{x}_{\backslash t} = [x_1, ..., x_{t-1}, x_{t+1}, ..., x_n]$. Typically the context is represented as the input sequence with x_t replaced by a special `[MASK]` placeholder token. This masked sequence is encoded into vector representations by a transformer network (Vaswani et al., 2017). Then the representation at position t is passed into a softmax layer to produce a distribution over tokens $p_\theta(x_t|\boldsymbol{x}_{\backslash t})$ for the position.

2.1 The Electric Model

Electric also models $p_{\text{data}}(x_t|\boldsymbol{x}_{\backslash t})$, but does not use masking or a softmax layer. Electric first maps the unmasked input $\boldsymbol{x} = [x_1, ..., x_n]$ into contextualized vector representations $\boldsymbol{h}(\boldsymbol{x}) = [\boldsymbol{h}_1, ..., \boldsymbol{h}_n]$ using a transformer network. The model assigns a given position t an energy score

$$E(\boldsymbol{x})_t = \boldsymbol{w}^T \boldsymbol{h}(\boldsymbol{x})_t$$

using a learned weight vector $\boldsymbol{w}$. The energy function defines a distribution over the possible tokens at position t as

$$p_\theta(x_t|\boldsymbol{x}_{\backslash t}) = \exp\left(-E(\boldsymbol{x})_t\right)/Z(\boldsymbol{x}_{\backslash t})$$
$$= \frac{\exp\left(-E(\boldsymbol{x})_t\right)}{\sum_{x' \in \mathcal{V}} \exp\left(-E(\text{REPLACE}(\boldsymbol{x}, t, x'))_t\right)}$$

where $\text{REPLACE}(\boldsymbol{x}, t, x')$ denotes replacing the token at position t with x' and $\mathcal{V}$ is the vocabulary, in practice usually word pieces (Sennrich et al., 2016). Unlike with BERT, which produces the probabilities for all possible tokens x' using a softmax layer, a candidate x' is passed in as *input* to the transformer. As a result, computing p_θ is prohibitively expensive because the partition function $Z_\theta(\boldsymbol{x}_{\backslash t})$ requires running the transformer $|\mathcal{V}|$ times; unlike most EBMs, the intractability of $Z_\theta(\boldsymbol{x}_{\backslash t})$ is due to the expensive scoring function rather than having a large sample space.

2.2 NCE Loss

As computing the exact likelihood is intractable, training energy-based models such as Electric with standard maximum-likelihood estimation is not possible. Instead, we use (conditional) Noise-Contrastive Estimation (NCE) (Gutmann and Hyvärinen, 2010; Ma and Collins, 2018), which provides a way of efficiently training an un-normalized model that does not compute $Z_\theta(\boldsymbol{x}_{\backslash t})$. NCE learns the parameters of a model by defining a binary classification task where samples from the data distribution have to be distinguished from samples generated by a noise distribution $q(x_t|\boldsymbol{x}_{\backslash t})$. First, we define the un-normalized output

$$\hat{p}_\theta(x_t|\boldsymbol{x}_{\backslash t}) = \exp\left(-E(\boldsymbol{x})_t\right)$$

Operationally, NCE can be viewed as follows:

- A positive data point is a text sequence $\boldsymbol{x}$ from the data and position in the sequence t.
- A negative data point is the same except x_t, the token at position t, is replaced with a noise token $\hat{x}_t$ sampled from q.
- Define a binary classifier D that estimates the probability of a data point being positive as
$$\frac{n \cdot \hat{p}_\theta(x_t|\boldsymbol{x}_{\backslash t})}{n \cdot \hat{p}_\theta(x_t|\boldsymbol{x}_{\backslash t}) + k \cdot q(x_t|\boldsymbol{x}_{\backslash t})}$$
- The binary classifier is trained to distinguish positive vs negative data points, with k negatives sampled for every n positive data points.

Formally, the NCE loss $\mathcal{L}(\theta)$ is

$$n \cdot \mathop{\mathbb{E}}_{\boldsymbol{x},t}\left[-\log \frac{n \cdot \hat{p}_\theta(x_t|\boldsymbol{x}_{\backslash t})}{n \cdot \hat{p}_\theta(x_t|\boldsymbol{x}_{\backslash t}) + k \cdot q(x_t|\boldsymbol{x}_{\backslash t})}\right] +$$
$$k \cdot \mathop{\mathbb{E}}_{\substack{\boldsymbol{x},t \\ \hat{x}_t \sim q}}\left[-\log \frac{k \cdot q(\hat{x}_t|\boldsymbol{x}_{\backslash t})}{n \cdot \hat{p}_\theta(\hat{x}_t|\boldsymbol{x}_{\backslash t}) + k \cdot q(\hat{x}_t|\boldsymbol{x}_{\backslash t})}\right]$$

This loss is minimized when $\hat{p}_\theta$ matches the data distribution p_{data} (Gutmann and Hyvärinen, 2010). A consequence of this property is that the model learns to be self-normalized such that $Z_\theta(\boldsymbol{x}_{\setminus t}) = 1$.

2.3 Training Algorithm

To minimize the loss, the expectations could be approximated by sampling as shown in Algorithm 1. Taking the gradient of this estimated loss produces an unbiased estimate of $\nabla_\theta \mathcal{L}(\theta)$. However, this algorithm is computationally expensive to run, since it requires $k + 1$ forward passes through the transformer to compute the $\hat{p}_\theta$s (once for the positive samples and once for each negative sample). We propose a much more efficient approach that replaces k input tokens with noise samples *simultaneously* shown in Algorithm 2. It requires just one pass through the transformer for k noise samples and $n - k$ data samples. However, this procedure only truly minimizes $\mathcal{L}$ if $\hat{p}_\theta(x_t|\boldsymbol{x}_{\setminus t}) = \hat{p}_\theta(x_t|\boldsymbol{x}^{\text{noised}}_{\setminus t})$. To apply this efficiency trick we are making the assumption they are approximately equal, which we argue is reasonable because (1) we choose a small k of $\lceil 0.15n \rceil$ and (2) q is trained to be close to the data distribution (see below). This efficiency trick is analogous to BERT masking out multiple tokens per input sequence.

Algorithm 1 Naive NCE loss estimation

```
Given: Input sequence $\boldsymbol{x}$, number of negative samples $k$, noise distribution $q$, model $\hat{p}_\theta$.
1. Initialize the loss as $\sum_{t=1}^{n} \left( -\log \frac{n \cdot \hat{p}_\theta(x_t|\boldsymbol{x}_{\setminus t})}{n \cdot \hat{p}_\theta(x_t|\boldsymbol{x}_{\setminus t}) + k \cdot q(x_t|\boldsymbol{x}_{\setminus t})} \right)$.
2. Sample $k$ negative samples according to $t \sim \text{unif}\{1, n\}$, $\hat{x}_t \sim q(x_t|\boldsymbol{x}_{\setminus t})$.
3. For each negative sample, add to the loss $-\log \frac{k \cdot q(\hat{x}_t|\boldsymbol{x}_{\setminus t})}{n \cdot \hat{p}_\theta(\hat{x}_t|\boldsymbol{x}_{\setminus t}) + k \cdot q(\hat{x}_t|\boldsymbol{x}_{\setminus t})}$.
```

Algorithm 2 Efficient NCE loss estimation

```
Given: Input sequence $\boldsymbol{x}$, number of negative samples $k$, noise distribution $q$, model $\hat{p}_\theta$.
1. Pick $k$ unique random positions $R = \{r_1, ..., r_k\}$ where each $r_i$ is $1 \le r_i \le n$.
2. Replace the $k$ random positions with negative samples: $\hat{x}_i \sim q(x_i|\boldsymbol{x}_{\setminus i})$ for $i \in R$,
   $\boldsymbol{x}^{\text{noised}} = \text{REPLACE}(\hat{\boldsymbol{x}}, R, \hat{X})$.
3. For each position $t = 1$ to $n$: add to the loss
   $-\log \frac{k \cdot q(\hat{x}_t|\boldsymbol{x}_{\setminus t})}{(n-k) \cdot \hat{p}_\theta(\hat{x}_t|\boldsymbol{x}^{\text{noised}}_{\setminus t}) + k \cdot q(\hat{x}_t|\boldsymbol{x}_{\setminus t})}$   if $t \in R$
   $-\log \frac{(n-k) \cdot \hat{p}_\theta(x_t|\boldsymbol{x}^{\text{noised}}_{\setminus t})}{(n-k) \cdot \hat{p}_\theta(x_t|\boldsymbol{x}^{\text{noised}}_{\setminus t}) + k \cdot q(x_t|\boldsymbol{x}_{\setminus t})}$   otherwise
```

2.4 Noise Distribution

The noise distribution q comes from a neural network trained to match p_{data}. NCE commonly employs this idea to ensure the classification task is sufficiently challenging for the model (Gutmann and Hyvärinen, 2010; Wang and Ou, 2018). In particular, we use a two-tower cloze model as proposed by Baevski et al. (2019), which is more accurate than a language model because it uses context to both sides of each token. The model runs two transformers T_{LTR} and T_{RTL} over the input sequence. These transformers apply causal masking so one processes the sequence left-to-right and the other operates right-to-left. The model's predictions come from a softmax layer applied to the concatenated states of the two transformers:

$$\overrightarrow{\boldsymbol{h}} = T_{\text{LTR}}(\boldsymbol{x}), \quad \overleftarrow{\boldsymbol{h}} = T_{\text{RTL}}(\boldsymbol{x})$$
$$q(x_t|\boldsymbol{x}_{\setminus t}) = \text{softmax}(\boldsymbol{W}[\overrightarrow{\boldsymbol{h}}_{t-1}, \overleftarrow{\boldsymbol{h}}_{t+1}])_{x_t}$$

The noise distribution is trained simultaneously with Electric using standard maximum likelihood estimation over the data. The model producing the noise distribution is much smaller than Electric to reduce the computational overhead.

2.5 Connection to ELECTRA

Electric is closely related to the ELECTRA pre-training method. ELECTRA also trains a binary classifier (the "discriminator") to distinguish data tokens from noise tokens produced by a "generator" network. However, ELECTRA's classifier is simply a sigmoid layer on top of the transformer: it models the probability of a token being negative (i.e., as replaced by a noise sample) as $\sigma(E(\boldsymbol{x})_t)$ where σ denotes the sigmoid function. Electric on the other hand models this probability as

$$\frac{k \cdot q(x|\boldsymbol{x}_{\setminus t})}{n \cdot \exp\left(-E(\boldsymbol{x})_t\right) + k \cdot q(x|\boldsymbol{x}_{\setminus t})} = \sigma\left(E(\boldsymbol{x})_t + \log\left(\frac{k \cdot q(x|\boldsymbol{x}_{\setminus t})}{n}\right)\right)$$

While ELECTRA learns whether a token is more likely to come from the data distribution p_{data} or noise distribution q, Electric only learns p_{data} because q is passed into the model directly. This difference is analogous to using negative sampling (Mikolov et al., 2013) vs. noise-contrastive estimation (Mnih and Kavukcuoglu, 2013) for learning word embeddings.

Model	**MultiNLI**	**SQuAD 2.0**	**GLUE Avg.**
BERT	84.3	73.7	82.2
XLNet	85.8	78.5	–
ELECTRA	86.2	80.5	85.1
Electric	85.7	80.1	84.1

Table 1: Dev-set scores of pre-trained models on downstream tasks. To provide direct comparisons, we only show base-sized models pre-trained on WikiBooks.

A disadvantage of Electric compared to ELECTRA is that it is less flexible in the choice of noise distribution. Since ELECTRA's binary classifier does not need to access q, its q only needs to be defined for negative sample positions in the input sequence. Therefore ELECTRA can use a masked language model rather than a two-tower cloze model for q. An advantage of Electric is that it directly provides (un-normalized) probabilities $\hat{p}_\theta$ for tokens, making it useful for applications such as re-ranking the outputs of text generation systems. The differences between ELECTRA and Electric are summarized below:

Model	Noise Dist.	Binary Classifier
Electric	Two-Tower Cloze Model	$\sigma\left(E(\boldsymbol{x})_t + \log\left(\frac{k \cdot q(x\|\boldsymbol{x}_{\setminus t})}{n}\right)\right)$
ELECTRA	Masked LM	$\sigma(E(\boldsymbol{x})_t)$

3 Experiments

We train two Electric models the same size as BERT-Base (110M parameters): one on Wikipedia and BooksCorpus (Zhu et al., 2015) for comparison with BERT and one on OpenWebTextCorpus (Gokaslan and Cohen, 2019) for comparison[2] with GPT-2. The noise distribution transformers T_{LTR} and T_{RTL} are 1/4 the hidden size of Electric. We do no hyperparameter tuning, using the same hyperparameter values as ELECTRA. Further details on training are in the appendix.

3.1 Transfer to Downstream Tasks

We evaluate fine-tuning the Electric model on the GLUE natural language understanding benchmark (Wang et al., 2019) and the SQuAD 2.0 question answering dataset (Rajpurkar et al., 2018). We report exact-match for SQuAD, average score[3] over the GLUE tasks[4], and accuracy on the multi-genre natural language inference GLUE task. Reported scores are medians over 10 fine-tuning runs with different random seeds. We use the same fine-tuning setup and hyperparameters as ELECTRA.

Results are shown in Table 1. Electric scores better than BERT, showing the energy-based formulation improves cloze model pre-training. However, Electric scores slightly lower than ELECTRA. One possible explanation is that Electric's noise distribution is worse because a two-tower cloze model is less expressive than a masked LM. We tested this hypothesis by training ELECTRA with the same two-tower noise model as Electric. Performance did indeed go down, but it only explained about half the gap. The surprising drop in performance suggests that learning the difference between the data and generations from a low-capacity model leads to better representations than only learning the data distribution, but we believe further research is needed to fully understand the discrepancy.

3.2 Fast Pseudo-Log-Likelihood Scoring

An advantage of Electric over BERT is that it can efficiently produce pseudo-log-likelihood (PLL) scores for text (Wang and Cho, 2019). PLLs for Electric are

$$\text{PLL}(\boldsymbol{x}) = \sum_{t=1}^{n} \log(\hat{p}_\theta(x_t|\boldsymbol{x}_{\setminus t})) = \sum_{t=1}^{n} -E(\boldsymbol{x})_t$$

PLLs can be used to re-rank the outputs of an NMT or ASR system. While historically log-likelihoods from language models have been used for such reranking, recent work has demonstrated that PLLs from masked language models perform better (Shin et al., 2019). However, computing PLLs from a masked language model requires n passes of the transformer: once with each token masked out. Salazar et al. (2020) suggest distilling BERT into a model that uses no masking to avoid this cost, but this model considerably under-performed regular LMs in their experiments.

Electric can produce PLLs for all input tokens in a single pass like a LM while being bidirectional like a masked LM. We use the PLLs from Electric for re-ranking the 100-best hypotheses of a 5-layer BLSTMP model from ESPnet (Watanabe et al., 2018) on the 960-hour LibriSpeech corpus (Panayotov et al., 2015) following the same experimental setup and using the same n-best lists as Salazar

[2] The original GPT-2 dataset is not public, so we use a public re-implementation.

[3] Matthews correlation coefficient for CoLA, Spearman correlation for STS, accuracy for the other tasks.

[4] We exclude WNLI, for which models do not outperform the majority baseline.

et al. (2020). Given speech features $\boldsymbol{s}$ and speech recognition model f the re-ranked output is

$$\underset{\boldsymbol{x} \in \text{n-best}(f,\boldsymbol{s})}{\arg\max} \; f(\boldsymbol{x}|\boldsymbol{s}) + \lambda \text{PLL}(\boldsymbol{x})$$

where n-best$(f, \boldsymbol{s})$ consists of the top n (we use $n = 100$) predictions from the speech recognition model found with beam search, $f(\boldsymbol{x}|\boldsymbol{s})$ is the score the speech model assigns the candidate output sequence $\boldsymbol{x}$. We select the best λ on the dev set out of $[0.05, 0.1, ..., 0.95, 1.0]$, with different λs selected for the "clean" and "other" portions of the data.

We compare Electric against GPT-2 (Radford et al., 2019), BERT (Devlin et al., 2019), and two baseline systems that are bidirectional while only requiring a single transformer pass like Electric. TwoTower is a two-tower cloze model similar to Electric's noise distribution, but of the same size as Electric. ELECTRA-TT is identical to ELECTRA except it uses a two-tower noise distribution rather than a masked language model.[5] The noise distribution probabilities and binary classifiers scores of ELECTRA can be combined to assign probabilities for tokens as shown in Appendix G of the ELECTRA paper.

Results are shown in Table 2. Electric scores better than GPT-2 when trained on comparable data. While scoring worse than BERT, Electric is much faster to run. It also slightly outperforms ELECTRA-TT, which is consistent with the finding from Labeau and Allauzen (2018) that NCE outperforms negative sampling for training language models. Furthermore, Electric is simpler and faster than ELETRA-TT in that it does not require running the generator to produce PLL scores. TwoTower scores lower than Electric, presumably because it is not a "deeply" bidirectional model and instead just concatenates forward and backward hidden states.

4 Related Work

Language modeling (Dai and Le, 2015; Radford et al., 2018; Peters et al., 2018) and cloze modeling (Devlin et al., 2019; Baevski et al., 2019; Liu et al., 2019) have proven to be effective pre-training tasks for NLP. Unlike Electric, these methods follow the standard recipe of estimating token probabilities with an output softmax and using maximum-likelihood training.

Energy-based models have been widely explored in machine learning (Dayan et al., 1995; LeCun et al., 2007). While many training methods involve sampling from the EBM using gradient-based MCMC (Du and Mordatch, 2019) or Gibbs sampling (Hinton, 2002), we considered these approaches too slow for pre-training because they require multiple passes through the model per sample. We instead use noise-contrastive estimation (Gutmann and Hyvärinen, 2010), which has widely been used in NLP for learning word vectors (Mnih and Kavukcuoglu, 2013) and text generation models (Jean et al., 2014; Józefowicz et al., 2016). While EBMs have previously been applied to left-to-right (Wang et al., 2015) or globally normalized (Rosenfeld et al., 2001; Deng et al., 2020) text generation, they have not previously been applied to cloze models or for pre-training NLP models. Several papers have pointed out the connection between EBMs and GANs (Zhao et al., 2016; Finn et al., 2016), which is similar to the Electric/ELECTRA connection.

Model	Pre-train Data	Clean WER	Other WER	Runtime
None	–	7.26	20.37	0
BERT	WikiBooks	5.41	17.41	n
Electric	WikiBooks	5.65	17.42	1
GPT-2	OWT	5.64	17.60	1
TwoTower	OWT*	5.32	17.25	1
ELECTRA-TT	OWT*	5.22	17.01	1
Electric	OWT*	5.18	16.93	1

Table 2: Test-set word error rates on LibriSpeech after rescoring with base-sized models. None, GPT-2, and BERT results are from Salazar et al. (2020). Runtime is measured in passes through the transformer. "Clean" and "other" are easier and harder splits of the data. *We use a public re-implementation of OpenWebText.

5 Conclusion

We have developed an energy-based cloze model we call Electric and designed an efficient training algorithm for Electric based on noise-contrastive estimation. Although Electric can be derived solely from the cloze task, the resulting pre-training method is closely related to ELECTRA's GAN-like pre-training algorithm. While slightly underperforming ELECTRA on downstream tasks, Electric is useful for its ability to quickly produce pseudo-log-likelihood scores for text. Furthermore, it offers a clearer and more principled view of the ELECTRA objective as a "negative sampling" version of cloze pre-training.

[5] With ELECTRA's original masked LM generator, it would be impossible to score all tokens in a single pass.

Acknowledgements

We thank John Hewitt, Yuhao Zhang, Ashwin Paranjape, Sergey Levine, and the anonymous reviewers for their thoughtful comments and suggestions. Kevin is supported by a Google PhD Fellowship.

References

Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. 2019. Cloze-driven pretraining of self-attention networks. In *EMNLP*.

Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *SemEval@ACL*.

Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D. Manning, and Quoc V. Le. 2019. BAM! Born-again multi-task networks for natural language understanding. In *ACL*.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *NeurIPS*.

Peter Dayan, Geoffrey E. Hinton, Radford M. Neal, and Richard S. Zemel. 1995. The Helmholtz machine. *Neural Computation*, 7:889–904.

Yuntian Deng, Anton Bakhtin, Myle Ott, and Arthur Szlam. 2020. Residual energy-based models for text generation. In *ICLR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *IWP@IJCNLP*.

Yilun Du and Igor Mordatch. 2019. Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689*.

Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. 2016. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. In *NeurIPS 2016 Workshop on Adversarial Training*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B. Dolan. 2007. The third pascal recognizing textual entailment challenge. In *ACL-PASCAL@ACL*.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*.

Geoffrey E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. 2017. First Quora dataset release: Question pairs.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On using very large target vocabulary for neural machine translation. In *ACL*.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Matthieu Labeau and Alexandre Allauzen. 2018. Learning with noise-contrastive estimation: Easing training by learning to scale. In *COLING*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *ICLR*.

Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu Jie Huang. 2007. A tutorial on energy-based learning. In Gökhan Bakır, Thomas Hofmann, Bernhard Schölkopf, Alexander J. Smola, Ben Taskar, and S. V. N. Vishwanathan, editors, *Predicting Structured Data*, pages 191–246. MIT Press, Cambridge, MA.

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*.

Zhuang Ma and Michael Collins. 2018. Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency. In *EMNLP*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *NeurIPS*.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An asr corpus based on public domain audio books. *ICASSP*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on STILTs: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *https://blog.openai.com/language-unsupervised*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Technical Report.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *NAACL-HLT*.

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

Nils Reimers and Iryna Gurevych. 2018. Why comparing single performance scores does not allow to draw conclusions about machine learning approaches. *arXiv preprint arXiv:1803.09578*.

Ronald Rosenfeld, Stanley F. Chen, and Xiaojin Zhu. 2001. Whole-sentence exponential language models: A vehicle for linguistic-statistical integration. *Comput. Speech Lang.*, 15:55–73.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. Masked language model scoring. In *ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *NAACL-HLT*.

Joonbo Shin, Yoonhyung Lee, and Kyomin Jung. 2019. Effective sentence scoring method using bert for speech recognition. In *Asian Conference on Machine Learning*, pages 1081–1093.

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 *EMNLP*.

Wilson L. Taylor. 1953. "Cloze procedure": a new tool for measuring readability. *Journalism & Mass Communication Quarterly*, 30:415–433.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Alex Wang and Kyunghyun Cho. 2019. BERT has a mouth, and it must speak: BERT as a markov random field language model. *arXiv preprint arXiv:1902.04094*.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.

Bin Wang and Zhijian Ou. 2018. Learning neural trans-dimensional random field language models with noise-contrastive estimation. *ICASSP*.

Bin Wang, Zhijian Ou, and Zhiqiang Tan. 2015. Trans-dimensional random fields for language modeling. In *ACL*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, and Tsubasa Ochiai. 2018. Espnet: End-to-end speech processing toolkit. In *INTERSPEECH*.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*.

Junbo Zhao, Michael Mathieu, and Yann LeCun. 2016. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *ICCV*.

A Pre-Training Details

The neural architectures of our models are identical to BERT-Base (Devlin et al., 2019), although we believe incorporating additions such as relative position encodings (Shaw et al., 2018) would improve results. Our pre-training setup is the same as ELECTRA's (Clark et al., 2020), which adds some additional ideas from Liu et al. (2019) on top of the BERT codebase, such as dynamic masking and removing the next-sentence prediction task. We use the weight sharing trick from ELECTRA, where the transformers producing the proposal distribution and the main transformer share token embeddings. We do not use whole-word or n-gram masking, although we believe it would improve results too.

We did no hyperparameter tuning, directly using the hyperparameters from ELECTRA-Base for Electric and our baselines. These hyperparameters are slightly modified from the ones used in BERT; for completeness, we show these values in Table 3. The hidden sizes, feed-forward hidden sizes, and number of attention heads of the two transformers T_{LTR} and T_{RTL} used to produce the proposal distribution are 1/4 the size of Electric. We chose this value because it keeps the compute comparable to ELECTRA; running two 1/4-sized transformers takes roughly the same compute as running one 1/3-sized transformer, which is the size of ELECTRA's generator. To make the compute exactly equal, we train Electric for slightly fewer steps than ELECTRA. This same generator architecture was used for ELECTRA-TT. The TwoTower baseline consists of two transformers 2/3 the size of BERT's, which takes approximately the same compute to run. The Electric models, ELECTRA-Base, and BERT-Base all use the same amount of pre-train compute (e.g., Electric is trained for fewer steps than BERT due to the extra compute from the proposal distribution), which equates to approximately three days of training on 16 TPUv2s.

B Fine-Tuning Details

We use ELECTRA's top-level classifiers and hyperparameter values for fine-tuning as well. For GLUE tasks, a simple linear classifier is added on top of the pre-trained transformer. For SQuAD, a question answering module similar XLNet's (Yang et al., 2019) is added on top of the transformer, which is slightly more sophisticated than BERT's in that it jointly rather than independently predicts the start and end positions and has an "answerability" classifier added for SQuAD 2.0. ELECTRA's hyperparameters are similar to BERT's, with the main difference being the addition of a layer-wise learning rate decay where layers of the network closer to the output have a higher learning rate. Following BERT, we submit the best of 10 models fine-tuned with different random seeds to the GLUE leaderboard for test set results.

C Dataset Details

We provide details on the fine-tuning datasets below. All datasets are in English. GLUE data can be downloaded at `https://gluebenchmark.com/` and SQuAD data can be downloaded at `https://rajpurkar.github.io/SQuAD-explorer/`.

- **CoLA:** Corpus of Linguistic Acceptability (Warstadt et al., 2018). The task is to determine whether a given sentence is grammatical or not. The dataset contains 8.5k train examples from books and journal articles on linguistic theory.

- **SST:** Stanford Sentiment Treebank (Socher et al., 2013). The tasks is to determine if the sentence is positive or negative in sentiment. The dataset contains 67k train examples from movie reviews.

- **MRPC:** Microsoft Research Paraphrase Corpus (Dolan and Brockett, 2005). The task is to predict whether two sentences are semantically equivalent or not. The dataset contains 3.7k train examples from online news sources.

- **STS:** Semantic Textual Similarity (Cer et al., 2017). The tasks is to predict how semantically similar two sentences are on a 1-5 scale. The dataset contains 5.8k train examples drawn from new headlines, video and image captions, and natural language inference data.

- **QQP:** Quora Question Pairs (Iyer et al., 2017). The task is to determine whether a pair of questions are semantically equivalent. The dataset contains 364k train examples from the community question-answering website Quora.

- **MNLI:** Multi-genre Natural Language Inference (Williams et al., 2018). Given a premise sentence and a hypothesis sentence, the task is to predict whether the premise entails the hypothesis, contradicts the hypothesis, or neither. The dataset contains 393k train examples drawn from ten different sources.

Hyperparameter	**Pre-Training**	**Fine-Tuning**
Number of layers	12	
Hidden Size	768	
FFN inner hidden size	3072	
Attention heads	12	
Attention head size	64	
Embedding Size	768	
Proposal Transformer Size	1/4	NA
Negative sample percent	15	NA
Learning Rate Decay	Linear	
Warmup steps	10000	First 10%
Learning Rate	5e-4	1e-4
Layerwise LR decay	None	0.8
Adam ϵ	1e-6	
Adam β_1	0.9	
Adam β_2	0.999	
Attention Dropout	0.1	
Dropout	0.1	
Weight Decay	0.01	0
Batch Size	256	32
Train Steps	700K	10 epochs for RTE and STS 2 for SQuAD, 3 for other tasks

Table 3: Hyperparameters for Electric; the values are identical to ELECTRA's other than the train steps and different-sized proposal network (see text), but we include them here for completeness. If not shown, the fine-tuning hyperparameter is the same as the pre-training one.

- **QNLI:** Question Natural Language Inference; constructed from SQuAD (Rajpurkar et al., 2016). The task is to predict whether a context sentence contains the answer to a question sentence. The dataset contains 108k train examples from Wikipedia.

- **RTE:** Recognizing Textual Entailment (Giampiccolo et al., 2007). Given a premise sentence and a hypothesis sentence, the task is to predict whether the premise entails the hypothesis or not. The dataset contains 2.5k train examples from a series of annual textual entailment challenges.

- **SQuAD 1.1:** Stanford Question Answering Dataset (Rajpurkar et al., 2016). Given a context paragraph and a question, the task is to select the span of text in the paragraph answering the question. The dataset contains 88k train examples from Wikipedia.

- **SQuAD 2.0:** Stanford Question Answering Dataset version 2.0 (Rajpurkar et al., 2018). This task adds addition questions to SQuAD whose answer does not exist in the context; models have to recognize when these questions occur and not return an answer for them. The dataset contains 130k train examples,

We report Spearman correlation for STS, Matthews correlation coefficient (MCC) for CoLA, exact match for SQuAD, and accuracy for the other tasks. We use the provided evaluation script for SQuAD[6], scipy to compute Spearman scores[7], and sklearn to compute MCC[8]. We use the standard train/dev/test splits.

D Detailed Results

We show detailed results on GLUE and SQuAD in Table 4 and detailed results on LibriSpeech reranking in Table 5. Following BERT, we do not show results on the WNLI GLUE task, as it is difficult to beat even the majority classifier using a standard fine-tuning-as-classifier approach. We show dev rather than test results on GLUE in the main paper because they are more reliable; the performance of fine-tuned models varies substantially based on the random seed (Phang et al., 2018; Clark et al., 2019; Dodge et al., 2020), but GLUE only supports submitting a single model rather than getting a median score of multiple models. While

[6] https://worksheets.codalab.org/rest/bundles/0x6b567e1cf2e041ec80d7098f031c5c9e/contents/blob/

[7] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.spearmanr.html

[8] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews_corrcoef.html

Model	**CoLA**	**SST**	**MRPC**	**STS**	**QQP**	**MNLI**	**QNLI**	**RTE**	**SQuAD 1.1**	**SQuAD 2.0**
	MCC	Acc	Acc	Spear	Acc	Acc	Acc	Acc	EM	EM
Dev set results										
BERT	58.4	92.8	86.0	87.8	90.8	84.5	88.6	68.5	80.8	73.7
XLNet	–	93.4	–	–	–	85.8	–	–	–	78.5
ELECTRA	65.8	92.4	87.9	89.1	90.9	86.2	92.4	76.3	84.5	80.5
Electric	61.8	91.9	88.0	89.4	90.6	85.7	92.1	73.4	84.5	80.1
Test set results										
BERT	52.1	93.5	84.8	85.8	89.2	84.6	90.5	66.4	–	–
ELECTRA	59.7	93.4	86.7	87.7	89.1	85.8	92.7	73.1	–	–
Electric	61.5	93.2	85.4	86.9	89.2	85.2	91.8	67.3	–	–

Table 4: GLUE scores pre-trained models on downstream tasks. To provide direct comparisons, we only show base-sized models pre-trained on WikiBooks and fine-tuned with standard single-task training.

Rescoring Model	**Pre-Training Data**	**Dev**		**Test**		**Transformer Passes**
		clean	other	clean	other	
None	–	7.17	19.79	7.26	20.37	0
BERT	WikiBooks	5.17	16.44	5.41	17.41	n
Electric	Wikibooks	5.47	16.56	5.65	17.42	1
GPT-2	OpenWebText	5.39	16.81	5.64	17.61	1
TwoTower	OpenWebText	5.12	16.37	5.32	17.25	1
ELECTRA-TT	OpenWebText	5.05	16.27	5.22	17.01	1
Electric	OpenWebText	4.97	16.23	5.18	16.93	1

Table 5: Word error rates on LibriSpeech after rescoring with base-sized models. None, GPT-2, and BERT results are from Salazar et al. (2020). Runtime is measured in passes through the transformer and data indicates the pre-training dataset. “Clean” and “other” are easier and harder splits of the data. *We use a public re-implementation of OpenWebText.

using dev-set model selection to choose the test set submission may alleviate the high variance of fine-tuning to some extent, such model selection is still not sufficient for reliable comparisons between methods (Reimers and Gurevych, 2018).