# Decision Transformer: Reinforcement Learning via Sequence Modeling

**Lili Chen**[*,1], **Kevin Lu**[*,1], **Aravind Rajeswaran**[2], **Kimin Lee**[1],
**Aditya Grover**[2], **Michael Laskin**[1], **Pieter Abbeel**[1], **Aravind Srinivas**[†,1], **Igor Mordatch**[†,3]
[*]equal contribution [†]equal advising
[1]UC Berkeley [2]Facebook AI Research [3]Google Brain
{lilichen, kzl}@berkeley.edu

## Abstract

We introduce a framework that abstracts Reinforcement Learning (RL) as a sequence modeling problem. This allows us to draw upon the simplicity and scalability of the Transformer architecture, and associated advances in language modeling such as GPT-x and BERT. In particular, we present Decision Transformer, an architecture that casts the problem of RL as conditional sequence modeling. Unlike prior approaches to RL that fit value functions or compute policy gradients, Decision Transformer simply outputs the optimal actions by leveraging a causally masked Transformer. By conditioning an autoregressive model on the desired return (reward), past states, and actions, our Decision Transformer model can generate future actions that achieve the desired return. Despite its simplicity, Decision Transformer matches or exceeds the performance of state-of-the-art model-free offline RL baselines on Atari, OpenAI Gym, and Key-to-Door tasks.

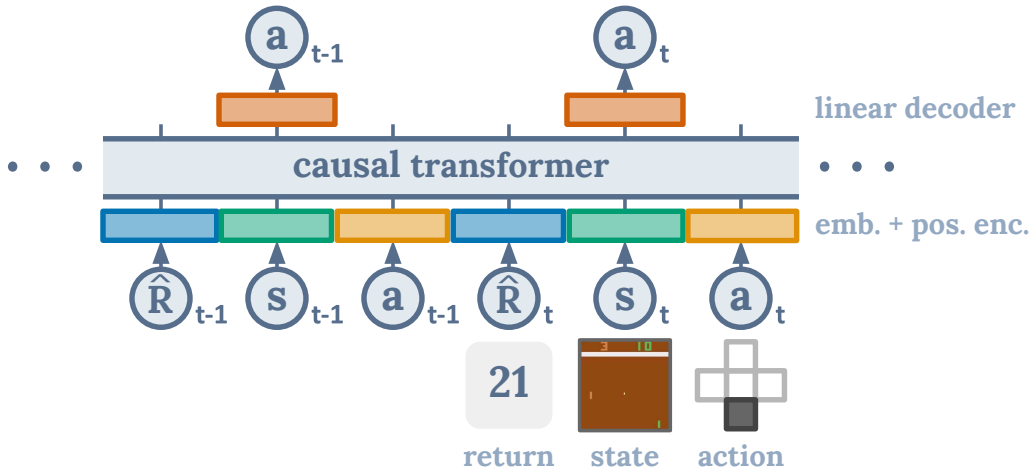


Figure 1: Decision Transformer architecture[1]. States, actions, and returns are fed into modality-specific linear embeddings and a positional episodic timestep encoding is added. Tokens are fed into a GPT architecture which predicts actions autoregressively using a causal self-attention mask.

[1]Our code is available at: https://sites.google.com/berkeley.edu/decision-transformer

## Contents

## 1 Introduction

Recent work has shown transformers [1] can model high-dimensional distributions of semantic concepts at scale, including effective zero-shot generalization in language [2] and out-of-distribution image generation [3]. Given the diversity of successful applications of such models, we seek to examine their application to sequential decision making problems formalized as reinforcement learning (RL). In contrast to prior work using transformers as an architectural choice for components within traditional RL algorithms [4, 5], we seek to study if generative trajectory modeling – i.e. modeling the joint distribution of the sequence of states, actions, and rewards – can serve as a *replacement* for conventional RL algorithms.

We consider the following shift in paradigm: instead of training a policy through conventional RL algorithms like temporal difference (TD) learning [6], we will train transformer models on collected experience using a sequence modeling objective. This will allow us to bypass the need for bootstrapping for long term credit assignment – thereby avoiding one of the "deadly triad" [6] known to destabilize RL. It also avoids the need for discounting future rewards, as typically done in TD learning, which can induce undesirable short-sighted behaviors. Additionally, we can make use of existing transformer frameworks widely used in language and vision that are easy to scale, utilizing a large body of work studying stable training of transformer models.

In addition to their demonstrated ability to model long sequences, transformers also have other advantages. Transformers can perform credit assignment directly via self-attention, in contrast to Bellman backups which slowly propagate rewards and are prone to "distractor" signals [7]. This can enable transformers to still work effectively in the presence of sparse or distracting rewards. Finally, empirical evidence suggest that a transformer modeling approach can model a wide distribution of behaviors, enabling better generalization and transfer [3].

We explore our hypothesis by considering *offline RL*, where we will task agents with learning policies from suboptimal data – producing maximally effective behavior from fixed, limited experience. This task is traditionally challenging due to error propagation and value overestimation [8]. However, it is a natural task when training with a sequence modeling objective. By training an autoregressive model on sequences of states, actions, and returns, we reduce policy sampling to autoregressive generative modeling. We can specify the expertise of the policy – which "skill" to query – by selecting the desired return tokens, acting as a prompt for generation.

**Illustrative example.** To get an intuition for our proposal, consider the task of finding the shortest path on a directed graph, which can be posed as an RL problem. The reward is $0$ when the agent is at the goal node and $-1$ otherwise. We train a GPT [9] model to predict next token in a sequence of returns-to-go (sum of future rewards), states, and actions. Training only on random walk data – with no expert demonstrations – we can generate *optimal* trajectories at test time by adding a prior to generate highest possible returns (see more details and empirical results in the Appendix) and subsequently generate the corresponding sequence of actions via conditioning. Thus, by combining the tools of sequence modeling with hindsight return information, we achieve policy improvement without the need for dynamic programming.

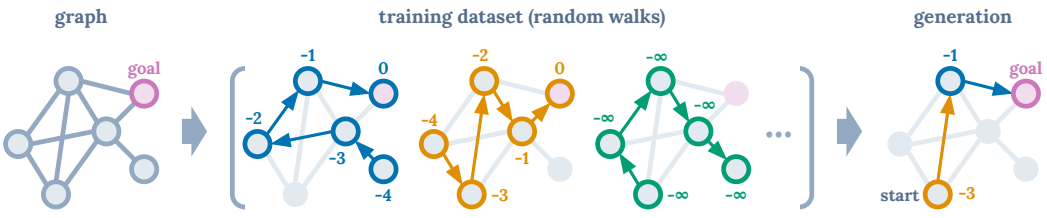


Figure 2: Illustrative example of finding shortest path for a fixed graph (left) posed as reinforcement learning. Training dataset consists of random walk trajectories and their per-node returns-to-go (middle). Conditioned on a starting state and generating largest possible return at each node, Decision Transformer sequences optimal paths.

Motivated by this observation, we propose Decision Transformer, where we use the GPT architecture to autoregressively model trajectories (shown in Figure 1). We study whether sequence modeling can perform policy optimization by evaluating Decision Transformer on offline RL benchmarks in Atari [10], OpenAI Gym [11], and Key-to-Door [12] environments. We show that – *without using dynamic programming* – Decision Transformer matches or exceeds the performance of state-of-the-art model-free offline RL algorithms [13, 14]. Furthermore, in tasks where long-term credit assignment is required, Decision Transformer capably outperforms the RL baselines. With this work, we aim to bridge sequence modeling and transformers with RL, and hope that sequence modeling serves as a strong algorithmic paradigm for RL.

## 2 Preliminaries

### 2.1 Offline reinforcement learning

We consider learning in a Markov decision process (MDP) described by the tuple ($\mathcal{S}$, $\mathcal{A}$, $P$, $\mathcal{R}$). The MDP tuple consists of states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, transition dynamics $P(s'|s,a)$, and a reward function $r = \mathcal{R}(s,a)$. We use $s_t$, $a_t$, and $r_t = \mathcal{R}(s_t, a_t)$ to denote the state, action, and reward at timestep $t$, respectively. A trajectory is made up of a sequence of states, actions, and rewards: $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T, a_T, r_T)$. The return of a trajectory at timestep $t$, $R_t = \sum_{t'=t}^{T} r_{t'}$, is the sum of future rewards from that timestep. The goal in reinforcement learning is to learn a policy which maximizes the expected return $\mathbb{E}\left[\sum_{t=1}^{T} r_t\right]$ in an MDP. In offline reinforcement learning, instead of obtaining data via environment interactions, we only have access to some fixed limited dataset consisting of trajectory rollouts of arbitrary policies. This setting is harder as it removes the ability for agents to explore the environment and collect additional feedback.

### 2.2 Transformers

Transformers were proposed by Vaswani et al. [1] as an architecture to efficiently model sequential data. These models consist of stacked self-attention layers with residual connections. Each self-attention layer receives $n$ embeddings $\{x_i\}_{i=1}^{n}$ corresponding to unique input tokens, and outputs $n$ embeddings $\{z_i\}_{i=1}^{n}$, preserving the input dimensions. The $i$-th token is mapped via linear transformations to a key $k_i$, query $q_i$, and value $v_i$. The $i$-th output of the self-attention layer is given by weighting the values $v_j$ by the normalized dot product between the query $q_i$ and other keys $k_j$:

$$z_i = \sum_{j=1}^{n} \texttt{softmax}(\{\langle q_i, k_{j'} \rangle\}_{j'=1}^{n})_j \cdot v_j. \tag{1}$$

As we shall see later, this allows the layer to assign "credit" by implicitly forming state-return associations via similarity of the query and key vectors (maximizing the dot product). In this work, we use the GPT architecture [9], which modifies the transformer architecture with a causal self-attention mask to enable autoregressive generation, replacing the summation/softmax over the $n$ tokens with only the previous tokens in the sequence ($j \in [1, i]$). We defer the other architecture details to the original papers.

## 3 Method

In this section, we present Decision Transformer, which models trajectories autoregressively with minimal modification to the transformer architecture, as summarized in Figure 1 and Algorithm 1.

**Trajectory representation.** The key desiderata in our choice of trajectory representation are that it should enable transformers to learn meaningful patterns and we should be able to conditionally generate actions at test time. It is nontrivial to model rewards since we would like the model to generate actions based on *future* desired returns, rather than past rewards. As a result, instead of feeding the rewards directly, we feed the model with the returns-to-go $\widehat{R}_t = \sum_{t'=t}^{T} r_{t'}$. This leads to the following trajectory representation which is amenable to autoregressive training and generation:

$$\tau = \left(\widehat{R}_1, s_1, a_1, \widehat{R}_2, s_2, a_2, \ldots, \widehat{R}_T, s_T, a_T\right). \tag{2}$$

At test time, we can specify the desired performance (e.g. 1 for success or 0 for failure), as well as the environment starting state, as the conditioning information to initiate generation. After executing the generated action for the current state, we decrement the target return by the achieved reward and repeat until episode termination.

**Architecture.** We feed the last $K$ timesteps into Decision Transformer, for a total of $3K$ tokens (one for each modality: return-to-go, state, or action). To obtain token embeddings, we learn a linear layer for each modality, which projects raw inputs to the embedding dimension, followed by layer normalization [15]. For environments with visual inputs, the state is fed into a convolutional encoder instead of a linear layer. Additionally, an embedding for each timestep is learned and added to each token – note this is different than the standard positional embedding used by transformers, as one timestep corresponds to three tokens. The tokens are then processed by a GPT [9] model, which predicts future action tokens via autoregressive modeling.

**Training.** We are given a dataset of offline trajectories. We sample minibatches of sequence length $K$ from the dataset. The prediction head corresponding to the input token $s_t$ is trained to predict $a_t$ – either with cross-entropy loss for discrete actions or mean-squared error for continuous actions – and the losses for each timestep are averaged. We did not find predicting the states or returns-to-go to improve performance, although it is easily permissible within our framework (as shown in Section 5.4) and would be an interesting study for future work.

**Algorithm 1** Decision Transformer Pseudocode (for continuous actions)

```
# R, s, a, t: returns-to-go, states, actions, or timesteps
# transformer: transformer with causal masking (GPT)
# embed_s, embed_a, embed_R: linear embedding layers
# embed_t: learned episode positional embedding
# pred_a: linear action prediction layer

# main model
def DecisionTransformer(R, s, a, t):
    # compute embeddings for tokens
    pos_embedding = embed_t(t)  # per-timestep (note: not per-token)
    s_embedding = embed_s(s) + pos_embedding
    a_embedding = embed_a(a) + pos_embedding
    R_embedding = embed_R(R) + pos_embedding

    # interleave tokens as (R_1, s_1, a_1, ..., R_K, s_K)
    input_embeds = stack(R_embedding, s_embedding, a_embedding)

    # use transformer to get hidden states
    hidden_states = transformer(input_embeds=input_embeds)

    # select hidden states for action prediction tokens
    a_hidden = unstack(hidden_states).actions

    # predict action
    return pred_a(a_hidden)

# training loop
for (R, s, a, t) in dataloader:  # dims: (batch_size, K, dim)
    a_preds = DecisionTransformer(R, s, a, t)
    loss = mean((a_preds - a)**2)  # L2 loss for continuous actions
    optimizer.zero_grad(); loss.backward(); optimizer.step()

# evaluation loop
target_return = 1  # for instance, expert-level return
R, s, a, t, done = [target_return], [env.reset()], [], [1], False
while not done:  # autoregressive generation/sampling
    # sample next action
    action = DecisionTransformer(R, s, a, t)[-1]  # for cts actions
    new_s, r, done, _ = env.step(action)

    # append new tokens to sequence
    R = R + [R[-1] - r]  # decrement returns-to-go with reward
    s, a, t = s + [new_s], a + [action], t + [len(R)]
    R, s, a, t = R[-K:], ...  # only keep context length of K
```

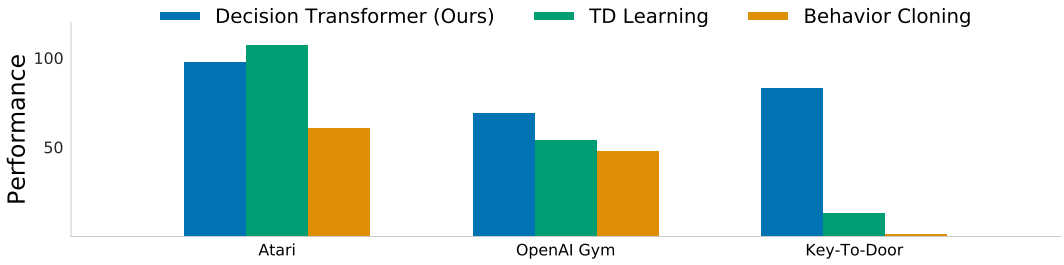


Figure 3: Results comparing Decision Transformer (ours) to TD learning (CQL) and behavior cloning across Atari, OpenAI Gym, and Minigrid. On a diverse set of tasks, Decision Transformer performs comparably or better than traditional approaches. Performance is measured by normalized episode return (see text for details).

## 4 Evaluations on Offline RL Benchmarks

In this section, we investigate the performance of Decision Transformer relative to dedicated offline RL and imitation learning algorithms. In particular, our primary points of comparison are model-free offline RL algorithms based on TD-learning, since our Decision Transformer architecture is fundamentally model-free in nature as well. Furthermore, TD-learning is the dominant paradigm in RL for sample efficiency, and also features prominently as a sub-routine in many model-based RL algorithms [16, 17]. We also compare with behavior cloning and variants, since it also involves a likelihood based policy learning formulation similar to ours. The exact algorithms depend on the environment but our motivations are as follows:

- **TD learning**: most of these methods use an action-space constraint or value pessimism, and will be the most faithful comparison to Decision Transformer, representing standard RL methods. A state-of-the-art model-free method is Conservative Q-Learning (CQL) [14] which serves as our primary comparison. In addition, we also compare against other prior model-free RL algorithms like BEAR [18] and BRAC [19].
- **Imitation learning**: this regime similarly uses supervised losses for training, rather than Bellman backups. We use behavior cloning here, and include a more detailed discussion in Section 5.1.

We evaluate on both discrete (Atari [10]) and continuous (OpenAI Gym [11]) control tasks. The former involves high-dimensional observation spaces and requires long-term credit assignment, while the latter requires fine-grained continuous control, representing a diverse set of tasks. Our main results are summarized in Figure 3, where we show averaged normalized performance for each domain.

### 4.1 Atari

The Atari benchmark [10] is challenging due to its high-dimensional visual inputs and difficulty of credit assignment arising from the delay between actions and resulting rewards. We evaluate our method on 1% of all samples in the DQN-replay dataset as per Agarwal et al. [13], representing 500 thousand of the 50 million transitions observed by an online DQN agent [20] during training; we report the mean and standard deviation of 3 seeds. We normalize scores based on a professional gamer, following the protocol of Hafner et al. [21], where 100 represents the professional gamer score and 0 represents a random policy.

We compare to CQL [14], REM [13], and QR-DQN [22] on four Atari tasks (Breakout, Qbert, Pong, and Seaquest) that are evaluated in Agarwal et al. [13]. We use context lengths of $K = 30$ for Decision Transformer (except $K = 50$ for Pong). We also report the performance of behavior cloning (BC), which utilizes the same network architecture and hyperparameters as Decision Transformer but does not have return-to-go conditioning[2]. For CQL, REM, and QR-DQN baselines, we report numbers directly from the CQL and REM papers. We show results in Table 1. Our method is competitive with CQL in 3 out of 4 games and outperforms or matches REM, QR-DQN, and BC on all 4 games.

[2]We also tried using an MLP with $K = 1$ as in prior work, but found this was worse than the transformer.

| Game | DT (Ours) | CQL | QR-DQN | REM | BC |
|---|---|---|---|---|---|
| Breakout | $\mathbf{267.5 \pm 97.5}$ | 211.1 | 17.1 | 8.9 | $138.9 \pm 61.7$ |
| Qbert | $15.4 \pm 11.4$ | $\mathbf{104.2}$ | 0.0 | 0.0 | $17.3 \pm 14.7$ |
| Pong | $106.1 \pm 8.1$ | $\mathbf{111.9}$ | 18.0 | 0.5 | $85.2 \pm 20.0$ |
| Seaquest | $\mathbf{2.5 \pm 0.4}$ | 1.7 | 0.4 | 0.7 | $2.1 \pm 0.3$ |

Table 1: Gamer-normalized scores for the 1% DQN-replay Atari dataset. We report the mean and variance across 3 seeds. Best mean scores are highlighted in bold. Decision Transformer (DT) performs comparably to CQL on 3 out of 4 games, and outperforms other baselines in most games.

### 4.2 OpenAI Gym

In this section, we consider the continuous control tasks from the D4RL benchmark [23]. We also consider a 2D reacher environment that is not part of the benchmark, and generate the datasets using a similar methodology to the D4RL benchmark. Reacher is a goal-conditioned task and has sparse rewards, so it represents a different setting than the standard locomotion environments (HalfCheetah, Hopper, and Walker). The different dataset settings are described below.

1. Medium: 1 million timesteps generated by a "medium" policy that achieves approximately one-third the score of an expert policy.

2. Medium-Replay: the replay buffer of an agent trained to the performance of a medium policy (approximately 25k-400k timesteps in our environments).

3. Medium-Expert: 1 million timesteps generated by the medium policy concatenated with 1 million timesteps generated by an expert policy.

We compare to CQL [14], BEAR [18], BRAC [19], and AWR [24]. CQL represents the state-of-the-art in model-free offline RL, an instantiation of TD learning with value pessimism. Score are normalized so that 100 represents an expert policy, as per Fu et al. [23]. CQL numbers are reported from the original paper; BC numbers are run by us; and the other methods are reported from the D4RL paper. Our results are shown in Table 2. Decision Transformer achieves the highest scores in a majority of the tasks and is competitive with the state of the art in the remaining tasks.

| Dataset | Environment | DT (Ours) | CQL | BEAR | BRAC-v | AWR | BC |
|---|---|---|---|---|---|---|---|
| Medium-Expert | HalfCheetah | $\mathbf{86.8 \pm 1.3}$ | 62.4 | 53.4 | 41.9 | 52.7 | 59.9 |
| Medium-Expert | Hopper | $107.6 \pm 1.8$ | $\mathbf{111.0}$ | 96.3 | 0.8 | 27.1 | 79.6 |
| Medium-Expert | Walker | $\mathbf{108.1 \pm 0.2}$ | 98.7 | 40.1 | 81.6 | 53.8 | 36.6 |
| Medium-Expert | Reacher | $\mathbf{89.1 \pm 1.3}$ | 30.6 | - | - | - | 73.3 |
| Medium | HalfCheetah | $42.6 \pm 0.1$ | 44.4 | 41.7 | $\mathbf{46.3}$ | 37.4 | 43.1 |
| Medium | Hopper | $\mathbf{67.6 \pm 1.0}$ | 58.0 | 52.1 | 31.1 | 35.9 | 63.9 |
| Medium | Walker | $74.0 \pm 1.4$ | 79.2 | 59.1 | $\mathbf{81.1}$ | 17.4 | 77.3 |
| Medium | Reacher | $\mathbf{51.2 \pm 3.4}$ | 26.0 | - | - | - | $\mathbf{48.9}$ |
| Medium-Replay | HalfCheetah | $36.6 \pm 0.8$ | 46.2 | 38.6 | $\mathbf{47.7}$ | 40.3 | 4.3 |
| Medium-Replay | Hopper | $\mathbf{82.7 \pm 7.0}$ | 48.6 | 33.7 | 0.6 | 28.4 | 27.6 |
| Medium-Replay | Walker | $\mathbf{66.6 \pm 3.0}$ | 26.7 | 19.2 | 0.9 | 15.5 | 36.9 |
| Medium-Replay | Reacher | $\mathbf{18.0 \pm 2.4}$ | $\mathbf{19.0}$ | - | - | - | 5.4 |
| **Average (Without Reacher)** | | $\mathbf{74.7}$ | 63.9 | 48.2 | 36.9 | 34.3 | 46.4 |
| **Average (All Settings)** | | $\mathbf{69.2}$ | 54.2 | - | - | - | 47.7 |

Table 2: Results for D4RL datasets[3]. We report the mean and variance for three seeds. Decision Transformer (DT) outperforms conventional RL algorithms on almost all tasks.

[3]Given that CQL is generally the strongest TD learning method, for Reacher we only run the CQL baseline.

# 5 Discussion

## 5.1 Does Decision Transformer perform behavior cloning on a subset of the data?

In this section, we seek to gain insight into whether Decision Transformer can be thought of as performing imitation learning on a subset of the data with a certain return. To investigate this, we propose a new method, Percentile Behavior Cloning (%BC), where we run behavior cloning on only the top $X\%$ of timesteps in the dataset, ordered by episode returns. The percentile $X\%$ interpolates between standard BC ($X = 100\%$) that trains on the entire dataset and only cloning the best observed trajectory ($X \to 0\%$), trading off between better generalization by training on more data with training a specialized model that focuses on a desirable subset of the data.

We show full results comparing %BC to Decision Transformer and CQL in Table 3, sweeping over $X \in [10\%, 25\%, 40\%, 100\%]$. Note that the only way to choose the optimal subset for cloning is to evaluate using rollouts from the environment, so %BC is not a realistic approach; rather, it serves to provide insight into the behavior of Decision Transformer. When data is plentiful – as in the D4RL regime – we find %BC can match or beat other offline RL methods. On most environments, Decision Transformer is competitive with the performance of the best %BC, indicating it can hone in on a particular subset after training on the entire dataset distribution.

| Dataset | Environment | DT (Ours) | 10%BC | 25%BC | 40%BC | 100%BC | CQL |
|---|---|---|---|---|---|---|---|
| Medium | HalfCheetah | $42.6 \pm 0.1$ | 42.9 | 43.0 | 43.1 | 43.1 | **44.4** |
| Medium | Hopper | $\mathbf{67.6 \pm 1.0}$ | 65.9 | 65.2 | 65.3 | 63.9 | 58.0 |
| Medium | Walker | $74.0 \pm 1.4$ | 78.8 | **80.9** | 78.8 | 77.3 | 79.2 |
| Medium | Reacher | $51.2 \pm 3.4$ | 51.0 | 48.9 | 58.2 | **58.4** | 26.0 |
| Medium-Replay | HalfCheetah | $36.6 \pm 0.8$ | 40.8 | 40.9 | 41.1 | 4.3 | **46.2** |
| Medium-Replay | Hopper | $\mathbf{82.7 \pm 7.0}$ | 70.6 | 58.6 | 31.0 | 27.6 | 48.6 |
| Medium-Replay | Walker | $66.6 \pm 3.0$ | **70.4** | 67.8 | 67.2 | 36.9 | 26.7 |
| Medium-Replay | Reacher | $18.0 \pm 2.4$ | **33.1** | 16.2 | 10.7 | 5.4 | 19.0 |
| **Average** | | 56.1 | **56.7** | 52.7 | 49.4 | 39.5 | 43.5 |

Table 3: Comparison between Decision Transformer (DT) and Percentile Behavior Cloning (%BC).

In contrast, when we study low data regimes – such as Atari, where we use 1% of a replay buffer as the dataset – %BC is weak (shown in Table 4). This suggests that in scenarios with relatively low amounts of data, Decision Transformer can outperform %BC by using all trajectories in the dataset to improve generalization, even if those trajectories are dissimilar from the return conditioning target. Our results indicate that Decision Transformer can be more effective than simply performing imitation learning on a subset of the dataset. On the tasks we considered, Decision Transformer either outperforms or is competitive to %BC, without the confound of having to select the optimal subset.

| Game | DT (Ours) | 10%BC | 25%BC | 40%BC | 100%BC |
|---|---|---|---|---|---|
| Breakout | $\mathbf{267.5 \pm 97.5}$ | $28.5 \pm 8.2$ | $73.5 \pm 6.4$ | $108.2 \pm 67.5$ | $138.9 \pm 61.7$ |
| Qbert | $15.4 \pm 11.4$ | $6.6 \pm 1.7$ | $16.0 \pm 13.8$ | $11.8 \pm 5.8$ | $\mathbf{17.3 \pm 14.7}$ |
| Pong | $\mathbf{106.1 \pm 8.1}$ | $2.5 \pm 0.2$ | $13.3 \pm 2.7$ | $72.7 \pm 13.3$ | $85.2 \pm 20.0$ |
| Seaquest | $\mathbf{2.5 \pm 0.4}$ | $1.1 \pm 0.2$ | $1.1 \pm 0.2$ | $1.6 \pm 0.4$ | $2.1 \pm 0.3$ |

Table 4: %BC scores for Atari. We report the mean and variance across 3 seeds. Decision Transformer (DT) outperforms all versions of %BC in most games.

## 5.2 How well does Decision Transformer model the distribution of returns?

We evaluate the ability of Decision Transformer to understand return-to-go tokens by varying the desired target return over a wide range – evaluating the multi-task distribution modeling capability of transformers. Figure 4 shows the average sampled return accumulated by the agent over the course of the evaluation episode for varying values of target return. On every task, the desired target returns and the true observed returns are highly correlated. On some tasks like Pong, HalfCheetah and Walker, Decision Transformer generates trajectories that almost perfectly match the desired returns

(as indicated by the overlap with the oracle line). Furthermore, on some Atari tasks like Seaquest, we can prompt the Decision Transformer with higher returns than the maximum episode return available in the dataset, demonstrating that Decision Transformer is sometimes capable of extrapolation.

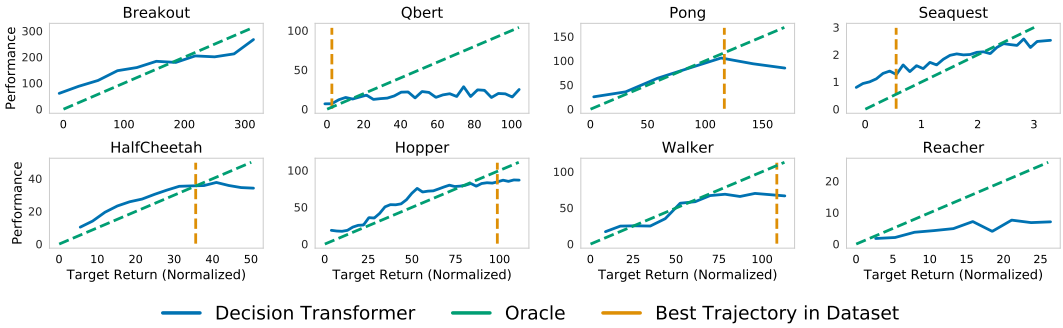


Figure 4: Sampled (evaluation) returns accumulated by Decision Transformer when conditioned on the specified target (desired) returns. **Top:** Atari. **Bottom:** D4RL medium-replay datasets.

### 5.3 What is the benefit of using a longer context length?

To assess the importance of access to previous states, actions, and returns, we ablate on the context length $K$. This is interesting since it is generally considered that the previous state (i.e. $K = 1$) is enough for reinforcement learning algorithms when frame stacking is used, as we do. Table 5 shows that performance of Decision Transformer is significantly worse when $K = 1$, indicating that past information is useful for Atari games. One hypothesis is that when we are representing a distribution of policies – like with sequence modeling – the context allows the transformer to identify which policy generated the actions, enabling better learning and/or improving the training dynamics.

| Game | DT (Ours) | DT with no context ($K = 1$) |
|---|---|---|
| Breakout | $\mathbf{267.5 \pm 97.5}$ | $73.9 \pm 10$ |
| Qbert | $\mathbf{15.1 \pm 11.4}$ | $13.6 \pm 11.3$ |
| Pong | $\mathbf{106.1 \pm 8.1}$ | $2.5 \pm 0.2$ |
| Seaquest | $\mathbf{2.5 \pm 0.4}$ | $0.6 \pm 0.1$ |

Table 5: Ablation on context length. Decision Transformer (DT) performs better when using a longer context length ($K = 50$ for Pong, $K = 30$ for others).

### 5.4 Does Decision Transformer perform effective long-term credit assignment?

To evaluate long-term credit assignment capabilities of our model, we consider a variant of the Key-to-Door environment proposed in Mesnard et al. [12]. This is a grid-based environment with a sequence of three phases: (1) in the first phase, the agent is placed in a room with a key; (2) then, the agent is placed in an empty room; (3) and finally, the agent is placed in a room with a door. The agent receives a binary reward when reaching the door in the third phase, but **only** if it picked up the key in the first phase. This problem is difficult for credit assignment because credit must be propagated from the beginning to the end of the episode, skipping over actions taken in the middle.

We train on datasets of trajectories generated by applying random actions and report success rates in Table 6. Furthermore, for the Key-to-Door environment we use the entire episode length as the context, rather than having a fixed content window as in the other environments. Methods that use highsight return information: our Decision Transformer model and %BC (trained only on successful episodes) are able to learn effective policies – producing near-optimal paths, despite only training on random walks. TD learning (CQL) cannot effectively propagate Q-values over the long horizons involved and gets poor performance.

| Dataset | DT (Ours) | CQL | BC | %BC | Random |
|---|---|---|---|---|---|
| 1K Random Trajectories | **71.8**% | 13.1% | 1.4% | 69.9% | 3.1% |
| 10K Random Trajectories | 94.6% | 13.3% | 1.6% | **95.1**% | 3.1% |

Table 6: Success rate for Key-to-Door environment. Methods using hindsight (Decision Transformer, %BC) can learn successful policies, while TD learning struggles to perform credit assignment.

### 5.5 Can transformers be accurate critics in sparse reward settings?

In previous sections, we established that decision transformer can produce effective policies (actors). We now evaluate whether transformer models can also be effective critics. We modify Decision Transformer to output return tokens in addition to action tokens on the Key-to-Door environment. Additionally, the first return token is not given, but it is predicted instead (i.e. the model learns the initial distribution $p(\hat{R}_1)$), similar to standard autoregressive generative models. We find that the transformer continuously updates reward probability based on events during the episode, shown in Figure 5 (Left). Furthermore, we find the transformer attends to critical events in the episode (picking up the key or reaching the door), shown in Figure 5 (Right), indicating formation of state-reward associations as discussed in Raposo et al. [25] and enabling accurate value prediction.

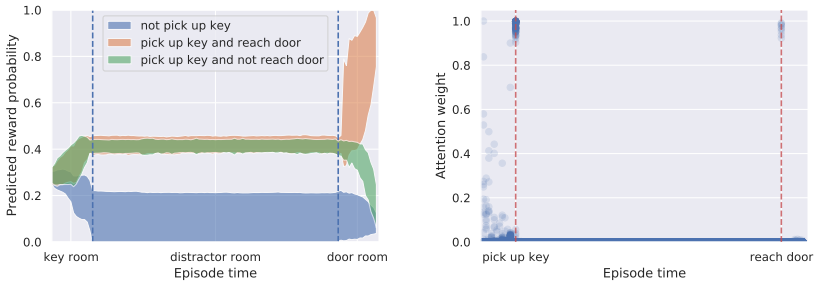


Figure 5: **Left:** Averages of running return probabilities predicted by the transformer model for three types of episode outcomes. **Right:** Transformer attention weights from all timesteps superimposed for a particular successful episode. The model attends to steps near pivotal events in the episode, such as picking up the key and reaching the door.

### 5.6 Does Decision Transformer perform well in sparse reward settings?

A known weakness of TD learning algorithms is that they require densely populated rewards in order to perform well, which can be unrealistic and/or expensive. In contrast, Decision Transformer can improve robustness in these settings since it makes minimal assumptions on the density of the reward. To evaluate this, we consider a delayed return version of the D4RL benchmarks where the agent does not receive any rewards along the trajectory, and instead receives the cumulative reward of the trajectory in the final timestep. Our results for delayed returns are shown in Table 7. Delayed returns minimally affect Decision Transformer; and due to the nature of the training process, while imitation learning methods are reward agnostic. While TD learning collapses, Decision Transformer and %BC still perform well, indicating that Decision Transformer can be more robust to delayed rewards.

| | | Delayed (Sparse) | | Agnostic | | Original (Dense) | |
|---|---|---|---|---|---|---|---|
| **Dataset** | **Environment** | **DT (Ours)** | **CQL** | **BC** | **%BC** | **DT (Ours)** | **CQL** |
| Medium-Expert | Hopper | $\mathbf{107.3 \pm 3.5}$ | 9.0 | 59.9 | 102.6 | 107.6 | 111.0 |
| Medium | Hopper | $60.7 \pm 4.5$ | 5.2 | 63.9 | **65.9** | 67.6 | 58.0 |
| Medium-Replay | Hopper | $\mathbf{78.5 \pm 3.7}$ | 2.0 | 27.6 | 70.6 | 82.7 | 48.6 |

Table 7: Results for D4RL datasets with delayed (sparse) reward. Decision Transformer (DT) and imitation learning are minimally affected by the removal of dense rewards, while CQL fails.

### 5.7 Why does Decision Transformer avoid the need for value pessimism or behavior regularization?

One key difference between Decision Transformer and prior offline RL algorithms is that we do not require policy regularization or conservatism to achieve good performance. Our conjecture is that TD-learning based algorithms learn an approximate value function and improve the policy by optimizing this value function. This act of optimizing a learned function can exacerbate and exploit any inaccuracies in the value function approximation, causing failures in policy improvement. Since Decision Transformer does not require explicit optimization using learned functions as objectives, it avoids the need for regularization or conservatism.

### 5.8 How can Decision Transformer benefit online RL regimes?

Offline RL and the ability to model behaviors has the potential to enable sample-efficient online RL for downstream tasks. Works studying the transition from offline to online generally find that likelihood-based approaches, like our sequence modeling objective, are more successful [26, 27]. As a result, although we studied offline RL in this work, we believe Decision Transformer can meaningfully improve online RL methods by serving as a strong model for behavior generation. For instance, Decision Transformer can serve as a powerful "memorization engine" and in conjunction with powerful exploration algorithms like Go-Explore [28], has the potential to simultaneously model and generative a diverse set of behaviors.

## 6 Related Work

### 6.1 Offline reinforcement learning

To mitigate the impact of distribution shift in offline RL, prior algorithms either (a) constrain the policy action space [29, 30, 31] or (b) incorporate value pessimism [29, 14], or (c) incorporate pessimism into learned dynamics models [32, 33]. Since we do not use Decision Transformers to explicitly learn the dynamics model, we primarily compare against model-free algorithms in our work; in particular, adding a dynamics model tends to improve the performance of model-free algorithms. Another line of work explores learning wide behavior distribution from an offline dataset by learning a task-agnostic set of skills, either with likelihood-based approaches [34, 35, 36, 37] or by maximizing mutual information [38, 39, 40]. Our work is similar to the likelihood-based approaches, which do not use iterative Bellman updates – although we use a simpler sequence modeling objective instead of a variational method, and use rewards for conditional generation of behaviors.

### 6.2 Supervised learning in reinforcement learning settings

Some prior methods for reinforcement learning bear more resemblance to static supervised learning, such as Q-learning [41, 42], which still uses iterative backups, or likelihood-based methods such as behavior cloning, which do not (discussed in previous section). Recent work [43, 44, 45] studies "upside-down" reinforcement learning (UDRL), which are similar to our method in seeking to model behaviors with a supervised loss conditioned on the target return. A key difference in our work is the shift of motivation to sequence modeling rather than supervised learning: while the practical methods differ primarily in the context length and architecture, sequence modeling enables behavior modeling even without access to the reward, in a similar style to language [9] or images [46], and is known to scale well [2]. The method proposed by Kumar et al. [44] is most similar to our method with $K = 1$, which we find sequence modeling/long contexts to outperform (see Section 5.3). Ghosh et al. [47] extends prior UDRL methods to use state goal conditioning, rather than rewards, and Paster et al. [48] further use an LSTM with state goal conditioning for goal-conditoned online RL settings.

Concurrent to our work, Janner et al. [49] propose Trajectory Transformer, which is similar to Decision Transformer but additionally uses state and return prediction, as well as discretization, which incorporates model-based components. We believe that their experiments, in addition to our results, highlight the potential for sequence modeling to be a generally applicable idea for reinforcement learning.

### 6.3 Credit assignment

Many works have studied better credit assignment via state-association, learning an architecture which decomposes the reward function such that certain "important" states comprise most of the credit [50, 51, 12]. They use the learned reward function to change the reward of an actor-critic algorithm to help propagate signal over long horizons. In particular, similar to our long-term setting, some works have specifically shown such state-associative architectures can perform better in delayed reward settings [52, 7, 53, 25]. In contrast, we allow these properties to naturally emerge in a transformer architecture, without having to explicitly learn a reward function or a critic.

### 6.4 Conditional language generation

Various works have studied guided generation for images [54] and language [55, 56]. Several works [57, 58, 59, 60, 61, 62] have explored training or fine-tuning of models for controllable text generation. Class-conditional language models can also be used to learn disciminators to guide generation [63, 55, 64, 65]. However, these approaches mostly assume constant "classes", while in reinforcement learning the reward signal is time-varying. Furthermore, it is more natural to prompt the model desired target return and continuously decrease it by the observed rewards over time, since the transformer model and environment jointly generate the trajectory.

### 6.5 Attention and transformer models

Transformers [1] have been applied successfully to many tasks in natural language processing [66, 9] and computer vision [67, 68]. However, transformers are relatively unstudied in RL, mostly due to differing nature of the problem, such as higher variance in training. Zambaldi et al. [5] showed that augmenting transformers with relational reasoning improve performance in combinatorial environments and Ritter et al. [69] showed iterative self-attention allowed for RL agents to better utilize episodic memories. Parisotto et al. [4] discussed design decisions for more stable training of transformers in the high-variance RL setting. Unlike our work, these still use actor-critic algorithms for optimization, focusing on novelty in architecture. Additionally, in imitation learning, some works have studied transformers as a replacement for LSTMs: Dasari and Gupta [70] study one-shot imitation learning, and Abramson et al. [71] combine language and image modalities for text-conditioned behavior generation.

## 7 Conclusion

We proposed Decision Transformer, seeking to unify ideas in language/sequence modeling and reinforcement learning. On standard offline RL benchmarks, we showed Decision Transformer can match or outperform strong algorithms designed explicitly for offline RL with minimal modifications from standard language modeling architectures.

We hope this work inspires more investigation into using large transformer models for RL. We used a simple supervised loss that was effective in our experiments, but applications to large-scale datasets could benefit from self-supervised pretraining tasks. In addition, one could consider more sophisticated embeddings for returns, states, and actions – for instance, conditioning on return distributions to model stochastic settings instead of deterministic returns. Transformer models can also be used to model the state evolution of trajectory, potentially serving as an alternative to model-based RL, and we hope to explore this in future work.

For real-world applications, it is important to understand the types of errors transformers make in MDP settings and possible negative consequences, which are underexplored. It will also be important to consider the datasets we train models on, which can potentially add destructive biases, particularly as we consider studying augmenting RL agents with more data which may come from questionable sources. For instance, reward design by nefarious actors can potentially generate unintended behaviors as our model generates behaviors by conditioning on desired returns.

## Acknowledgements

This research was supported by Berkeley Deep Drive, Open Philanthropy, and the National Science Foundation under NSF:NRI #2024675. Part of this work was completed when Aravind Rajeswaran was a PhD student at the University of Washington, where he was supported by the J.P. Morgan PhD Fellowship in AI (2020-21). We also thank Luke Metz and Daniel Freeman for valuable feedback and discussions, as well as Justin Fu for assistance in setting up D4RL benchmarks, and Aviral Kumar for assistance with the CQL baselines and hyperparameters.

## References


[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

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

[3] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092*, 2021.

[4] Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *International Conference on Machine Learning*, 2020.

[5] Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, et al. Deep reinforcement learning with relational inductive biases. In *International Conference on Learning Representations*, 2018.

[6] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT Press, 2018.

[7] Chia-Chun Hung, Timothy Lillicrap, Josh Abramson, Yan Wu, Mehdi Mirza, Federico Carnevale, Arun Ahuja, and Greg Wayne. Optimizing agent behavior over long time scales by transporting value. *Nature communications*, 10(1):1–12, 2019.

[8] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

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

[10] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[11] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[12] Thomas Mesnard, Théophane Weber, Fabio Viola, Shantanu Thakoor, Alaa Saade, Anna Harutyunyan, Will Dabney, Tom Stepleton, Nicolas Heess, Arthur Guez, et al. Counterfactual credit assignment in model-free reinforcement learning. *arXiv preprint arXiv:2011.09464*, 2020.

[13] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, 2020.

[14] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

[15] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[16] Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *ICML*, 1990.

[17] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, pages 12498–12509, 2019.

[18] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *arXiv preprint arXiv:1906.00949*, 2019.

[19] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[21] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[22] Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Conference on Artificial Intelligence*, 2018.

[23] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[24] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[25] David Raposo, Sam Ritter, Adam Santoro, Greg Wayne, Theophane Weber, Matt Botvinick, Hado van Hasselt, and Francis Song. Synthetic returns for long-term credit assignment. *arXiv preprint arXiv:2102.12425*, 2021.

[26] Yang Gao, Huazhe Xu, Ji Lin, Fisher Yu, Sergey Levine, and Trevor Darrell. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*, 2018.

[27] Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[28] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

[29] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, 2019.

[30] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, 2019.

[31] Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. In *International Conference on Learning Representations*, 2020.

[32] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

[33] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems*, 2020.

[34] Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. Opal: Offline primitive discovery for accelerating offline reinforcement learning. *arXiv preprint arXiv:2010.13611*, 2020.

[35] Víctor Campos, Alexander Trott, Caiming Xiong, Richard Socher, Xavier Giro-i Nieto, and Jordi Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International Conference on Machine Learning*, 2020.

[36] Karl Pertsch, Youngwoon Lee, and Joseph J Lim. Accelerating reinforcement learning with learned skill priors. *arXiv preprint arXiv:2010.11944*, 2020.

[37] Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. In *International Conference on Learning Representations*, 2021.

[38] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019.

[39] Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Reset-free lifelong learning with skill-space planning. *arXiv preprint arXiv:2012.03548*, 2020.

[40] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2020.

[41] Christopher Watkins. Learning from delayed rewards. 01 1989.

[42] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[43] Rupesh Kumar Srivastava, Pranav Shyam, Filipe Mutz, Wojciech Jaśkowski, and Jürgen Schmidhuber. Training agents using upside-down reinforcement learning. *arXiv preprint arXiv:1912.02877*, 2019.

[44] Aviral Kumar, Xue Bin Peng, and Sergey Levine. Reward-conditioned policies. *arXiv preprint arXiv:1912.13465*, 2019.

[45] Acting without rewards. 2019. URL `https://ogma.ai/2019/08/acting-without-rewards/`.

[46] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International Conference on Machine Learning*, pages 1691–1703. PMLR, 2020.

[47] Dibya Ghosh, Abhishek Gupta, Justin Fu, Ashwin Reddy, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals without reinforcement learning. *arXiv preprint arXiv:1912.06088*, 2019.

[48] Keiran Paster, Sheila A McIlraith, and Jimmy Ba. Planning from pixels using inverse dynamics models. *arXiv preprint arXiv:2012.02419*, 2020.

[49] Michael Janner, Qiyang Li, and Sergey Levine. Reinforcement learning as one big sequence modeling problem. *arXiv preprint arXiv:2106.02039*, 2021.

[50] Johan Ferret, Raphaël Marinier, Matthieu Geist, and Olivier Pietquin. Self-attentional credit assignment for transfer in reinforcement learning. *arXiv preprint arXiv:1907.08027*, 2019.

[51] Anna Harutyunyan, Will Dabney, Thomas Mesnard, Mohammad Azar, Bilal Piot, Nicolas Heess, Hado van Hasselt, Greg Wayne, Satinder Singh, Doina Precup, et al. Hindsight credit assignment. *arXiv preprint arXiv:1912.02503*, 2019.

[52] Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. *arXiv preprint arXiv:1806.07857*, 2018.

[53] Yang Liu, Yunan Luo, Yuanyi Zhong, Xi Chen, Qiang Liu, and Jian Peng. Sequence modeling of temporal credit assignment for episodic reinforcement learning. *arXiv preprint arXiv:1905.13420*, 2019.

[54] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Conference on Computer Vision and Pattern Recognition*, 2019.

[55] Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. Hafez: an interactive poetry generation system. In *Proceedings of ACL, System Demonstrations*, 2017.

[56] Lilian Weng. Controllable neural text generation. *lilianweng.github.io/lil-log*, 2021. URL `https://lilianweng.github.io/lil-log/2021/01/02/controllable-neural-text-generation.html`.

[57] Jessica Ficler and Yoav Goldberg. Controlling linguistic style aspects in neural language generation. *arXiv preprint arXiv:1707.02633*, 2017.

[58] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. Toward controlled generation of text. In *International Conference on Machine Learning*, 2017.

[59] Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. Explain yourself! leveraging language models for commonsense reasoning. *arXiv preprint arXiv:1906.02361*, 2019.

[60] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI conference on artificial intelligence*, 2017.

[61] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

[62] Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019.

[63] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*, 2019.

[64] Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. Learning to write with cooperative discriminators. *arXiv preprint arXiv:1805.06087*, 2018.

[65] Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. Gedi: Generative discriminator guided sequence generation. *arXiv preprint arXiv:2009.06367*, 2020.

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

[67] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, 2020.

[68] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[69] Sam Ritter, Ryan Faulkner, Laurent Sartran, Adam Santoro, Matt Botvinick, and David Raposo. Rapid task-solving in novel environments. *arXiv preprint arXiv:2006.03662*, 2020.

[70] Sudeep Dasari and Abhinav Gupta. Transformers for one-shot visual imitation. *arXiv preprint arXiv:2011.05970*, 2020.

[71] Josh Abramson, Arun Ahuja, Iain Barr, Arthur Brussee, Federico Carnevale, Mary Cassin, Rachita Chhaparia, Stephen Clark, Bogdan Damoc, Andrew Dudzik, et al. Imitating interactive intelligence. *arXiv preprint arXiv:2012.05672*, 2020.

[72] Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. Transformers: State-of-the-art natural language processing. In *Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020.

[73] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

# A Experimental Details

Code for experiments can be found in the supplementary material.

## A.1 Atari

We build our Decision Transformer implementation for Atari games off of minGPT (`https://github.com/karpathy/minGPT`), a publicly available re-implementation of GPT. We use most of the default hyperparameters from their character-level GPT example (`https://github.com/karpathy/minGPT/blob/master/play_char.ipynb`). We reduce the batch size (except in Pong), block size, number of layers, attention heads, and embedding dimension for faster training. For processing the observations, we use the DQN encoder from Mnih et al. [20] with an additional linear layer to project to the embedding dimension.

For return-to-go conditioning, we use either $1\times$ or $5\times$ the maximum return in the dataset, but more possibilities exist for principled return-to-go conditioning. In Atari experiments, we use Tanh instead of LayerNorm (as described in Section 3) after embedding each modality, but did this does not make a significant difference in performance. The full list of hyperparameters can be found in Table 8.

Table 8: Hyperparameters of DT (and %BC) for Atari experiments.

| **Hyperparameter** | **Value** |
|---|---|
| Number of layers | 6 |
| Number of attention heads | 8 |
| Embedding dimension | 128 |
| Batch size | 512 Pong<br>128 Breakout, Qbert, Seaquest |
| Context length $K$ | 50 Pong<br>30 Breakout, Qbert, Seaquest |
| Return-to-go conditioning | 90 Breakout ($\approx 1\times$ max in dataset)<br>2500 Qbert ($\approx 5\times$ max in dataset)<br>20 Pong ($\approx 1\times$ max in dataset)<br>1450 Seaquest ($\approx 5\times$ max in dataset) |
| Nonlinearity | ReLU, encoder<br>GeLU, otherwise |
| Encoder channels | $32, 64, 64$ |
| Encoder filter sizes | $8 \times 8, 4 \times 4, 3 \times 3$ |
| Encoder strides | $4, 2, 1$ |
| Max epochs | 5 |
| Dropout | 0.1 |
| Learning rate | $6 * 10^{-4}$ |
| Adam betas | $(0.9, 0.95)$ |
| Grad norm clip | 1.0 |
| Weight decay | 0.1 |
| Learning rate decay | Linear warmup and cosine decay (see code for details) |
| Warmup tokens | $512 * 20$ |
| Final tokens | $2 * 500000 * K$ |

## A.2 OpenAI Gym

### A.2.1 Decision Transformer

Our code is based on the Huggingface Transformers library [72]. Our hyperparameters on all OpenAI Gym tasks are shown below in Table 9. Heuristically, we find using larger models helps to model the distribution of returns, compared to standard RL model sizes (which learn one policy). For reacher we use a smaller context length than the other environments, which we find to be helpful as the environment is goal-conditioned and the episodes are shorter. We choose return targets based on expert performance for each environment, except for HalfCheetah where we find 50% performance to be better due to the datasets containing lower relative returns to the other environments. Models were trained for $10^5$ gradient steps using the AdamW optimizer [73] following PyTorch defaults.

Table 9: Hyperparameters of Decision Transformer for OpenAI Gym experiments.

| **Hyperparameter** | **Value** |
|---|---|
| Number of layers | 3 |
| Number of attention heads | 1 |
| Embedding dimension | 128 |
| Nonlinearity function | ReLU |
| Batch size | 64 |
| Context length $K$ | 20 HalfCheetah, Hopper, Walker<br>5 Reacher |
| Return-to-go conditioning | 6000 HalfCheetah<br>3600 Hopper<br>5000 Walker<br>50 Reacher |
| Dropout | 0.1 |
| Learning rate | $10^{-4}$ |
| Grad norm clip | 0.25 |
| Weight decay | $10^{-4}$ |
| Learning rate decay | Linear warmup for first $10^5$ training steps |

### A.2.2 Behavior Cloning

As briefly mentioned in Section 4.2, we found previously reported behavior cloning baselines to be weak, and so run them ourselves using a similar setup as Decision Transformer. We tried using a transformer architecture, but found using an MLP (as in previous work) to be stronger. We train for $2.5 \times 10^4$ gradient steps; training more did not improve performance. Other hyperparameters are shown in Table 10. The percentile behavior cloning experiments use the same hyperparameters.

Table 10: Hyperparameters of Behavior Cloning for OpenAI Gym experiments.

| **Hyperparameter** | **Value** |
|---|---|
| Number of layers | 3 |
| Embedding dimension | 256 |
| Nonlinearity function | ReLU |
| Batch size | 64 |
| Dropout | 0.1 |
| Learning rate | $10^{-4}$ |
| Weight decay | $10^{-4}$ |
| Learning rate decay | Linear warmup for first $10^5$ training steps |

## A.3 Graph Shortest Path

We give details of the illustrative example discussed in the introduction. The task is to find the shortest path on a fixed directed graph, which can be formulated as an MDP where reward is $0$ when the agent is at the goal node and $-1$ otherwise. The observation is the integer index of the graph node the agent is in. The action is the integer index of the graph node to move to next. The transition dynamics transport the agent to the action's node index if there is an edge in the graph, while the agent remains at the past node otherwise. The returns-to-go in this problem correspond to negative path lengths and maximizing them corresponds to generating shortest paths.

In this environment, we use the GPT model as described in Section 3 to generate both actions and return-to-go tokens. This makes it possible for the model it generate its own (realizable) returns-to-go $\hat{R}$. Since we require a return prompt to generate actions and we do assume knowledge of the optimal path length upfront, we use a simple prior over returns that favors shorter paths: $P_{\text{prior}}(\hat{R} = k) \propto T + 1 - k$, where $T$ is the maximum trajectory length. Then, it is combined with the return probabilities generated by the GPT model: $P(\hat{R}_t | s_{0:t}, a_{0:t-1}, \hat{R}_{0:t-1}) =$


Figure 6: Histogram of steps to reach the goal node for random walks on the graph, shortest possible paths to the goal, and attempted shortest paths generated by the transformer model. $\infty$ indicates the goal was not reached during the trajectory.

$P_{\text{GPT}}(\hat{R}_t|s_{0:t}, a_{0:t-1}, \hat{R}_{0:t-1}) \times P_{\text{prior}}(\hat{R}_t)^{10}$. Note that the prior and return-to-go predictions are entirely computable by the model, and thus avoids the need for any external or oracle information like the optimal path length. Adjustment of generation by a prior has also been used for similar purposes in controllable text generation in prior work [65].

We train on a dataset of $1,000$ graph random walk trajectories of $T = 10$ steps each with a random graph of $20$ nodes and edge sparsity coefficient of $0.1$. We report the results in Figure 6, where we find that transformer model is able to significantly improve upon the number of steps required to reach the goal, closely matching performance of optimal paths.

There are two reasons for the favorable performance on this task. In one case, the training dataset of random walk trajectories may contain a segment that directly corresponds to the desired shortest path, in which case it will be generated by the model. In the second case, generated paths are entirely original and are not subsets of trajectories in the training dataset - they are generated from stitching sub-optimal segments. We find this case accounts for $15.8\%$ of generated paths in the experiment.

While this is a simple example and uses a prior on generation that we do not use in other experiments for simplicity, it illustrates how hindsight return information can be used with generation priors to avoid the need for explicit dynamic programming.

## B Atari Task Scores

Table 11 shows the normalized scores used for normalization used in Hafner et al. [21]. Tables 12 and 13 show the raw scores corresponding to Tables 1 and 4, respectively. For %BC scores, we use the same hyperparameters as Decision Transformer for fair comparison. For REM and QR-DQN, there is a slight discrepancy between Agarwal et al. [13] and Kumar et al. [14]; we report raw data provided to us by REM authors.

| Game | Random | Gamer |
|---|---|---|
| Breakout | 2 | 30 |
| Qbert | 164 | 13455 |
| Pong | −21 | 15 |
| Seaquest | 68 | 42055 |

Table 11: Atari baseline scores used for normalization.

| Game | DT (Ours) | CQL | QR-DQN | REM | BC |
|---|---|---|---|---|---|
| Breakout | $\mathbf{76.9 \pm 27.3}$ | $61.1$ | $6.8$ | $4.5$ | $40.9 \pm 17.3$ |
| Qbert | $2215.8 \pm 1523.7$ | $\mathbf{14012.0}$ | $156.0$ | $160.1$ | $2464.1 \pm 1948.2$ |
| Pong | $17.1 \pm 2.9$ | $\mathbf{19.3}$ | $-14.5$ | $-20.8$ | $9.7 \pm 7.2$ |
| Seaquest | $\mathbf{1129.3 \pm 189.0}$ | $779.4$ | $250.1$ | $370.5$ | $968.6 \pm 133.8$ |

Table 12: Raw scores for the 1% DQN-replay Atari dataset. We report the mean and variance across 3 seeds. Best mean scores are highlighted in bold. Decision Transformer performs comparably to CQL on 3 out of 4 games, and usually outperforms other baselines.

| Game | DT (Ours) | 10%BC | 25%BC | 40%BC | 100%BC |
|---|---|---|---|---|---|
| Breakout | $\mathbf{76.9 \pm 27.3}$ | $10.0 \pm 2.3$ | $22.6 \pm 1.8$ | $32.3 \pm 18.9$ | $40.9 \pm 17.3$ |
| Qbert | $2215.8 \pm 1523.7$ | $1045 \pm 232.0$ | $2302.5 \pm 1844.1$ | $1674.1 \pm 776.0$ | $\mathbf{2464.1 \pm 1948.2}$ |
| Pong | $\mathbf{17.1 \pm 2.9}$ | $-20.3 \pm 0.1$ | $-16.2 \pm 1.0$ | $5.2 \pm 4.8$ | $9.7 \pm 7.2$ |
| Seaquest | $\mathbf{1129.3 \pm 189.0}$ | $521.3 \pm 103.0$ | $549.3 \pm 96.2$ | $758 \pm 169.1$ | $968.6 \pm 133.8$ |

Table 13: %BC scores for Atari. We report the mean and variance across 3 seeds. Decision Transformer usually outperforms %BC.

# Improving Language Understanding by Generative Pre-Training

**Alec Radford**
OpenAI
alec@openai.com

**Karthik Narasimhan**
OpenAI
karthikn@openai.com

**Tim Salimans**
OpenAI
tim@openai.com

**Ilya Sutskever**
OpenAI
ilyasu@openai.com

## Abstract

Natural language understanding comprises a wide range of diverse tasks such as textual entailment, question answering, semantic similarity assessment, and document classification. Although large unlabeled text corpora are abundant, labeled data for learning these specific tasks is scarce, making it challenging for discriminatively trained models to perform adequately. We demonstrate that large gains on these tasks can be realized by *generative pre-training* of a language model on a diverse corpus of unlabeled text, followed by *discriminative fine-tuning* on each specific task. In contrast to previous approaches, we make use of task-aware input transformations during fine-tuning to achieve effective transfer while requiring minimal changes to the model architecture. We demonstrate the effectiveness of our approach on a wide range of benchmarks for natural language understanding. Our general task-agnostic model outperforms discriminatively trained models that use architectures specifically crafted for each task, significantly improving upon the state of the art in 9 out of the 12 tasks studied. For instance, we achieve absolute improvements of 8.9% on commonsense reasoning (Stories Cloze Test), 5.7% on question answering (RACE), and 1.5% on textual entailment (MultiNLI).

## 1 Introduction

The ability to learn effectively from raw text is crucial to alleviating the dependence on supervised learning in natural language processing (NLP). Most deep learning methods require substantial amounts of manually labeled data, which restricts their applicability in many domains that suffer from a dearth of annotated resources [61]. In these situations, models that can leverage linguistic information from unlabeled data provide a valuable alternative to gathering more annotation, which can be time-consuming and expensive. Further, even in cases where considerable supervision is available, learning good representations in an unsupervised fashion can provide a significant performance boost. The most compelling evidence for this so far has been the extensive use of pre-trained word embeddings [10, 39, 42] to improve performance on a range of NLP tasks [8, 11, 26, 45].

Leveraging more than word-level information from unlabeled text, however, is challenging for two main reasons. First, it is unclear what type of optimization objectives are most effective at learning text representations that are useful for transfer. Recent research has looked at various objectives such as language modeling [44], machine translation [38], and discourse coherence [22], with each method outperforming the others on different tasks.[1] Second, there is no consensus on the most effective way to transfer these learned representations to the target task. Existing techniques involve a combination of making task-specific changes to the model architecture [43, 44], using intricate learning schemes [21] and adding auxiliary learning objectives [50]. These uncertainties have made it difficult to develop effective semi-supervised learning approaches for language processing.

[1] https://gluebenchmark.com/leaderboard

Preprint. Work in progress.

In this paper, we explore a semi-supervised approach for language understanding tasks using a combination of unsupervised pre-training and supervised fine-tuning. Our goal is to learn a universal representation that transfers with little adaptation to a wide range of tasks. We assume access to a large corpus of unlabeled text and several datasets with manually annotated training examples (target tasks). Our setup does not require these target tasks to be in the same domain as the unlabeled corpus. We employ a two-stage training procedure. First, we use a language modeling objective on the unlabeled data to learn the initial parameters of a neural network model. Subsequently, we adapt these parameters to a target task using the corresponding supervised objective.

For our model architecture, we use the *Transformer* [62], which has been shown to perform strongly on various tasks such as machine translation [62], document generation [34], and syntactic parsing [29]. This model choice provides us with a more structured memory for handling long-term dependencies in text, compared to alternatives like recurrent networks, resulting in robust transfer performance across diverse tasks. During transfer, we utilize task-specific input adaptations derived from traversal-style approaches [52], which process structured text input as a single contiguous sequence of tokens. As we demonstrate in our experiments, these adaptations enable us to fine-tune effectively with minimal changes to the architecture of the pre-trained model.

We evaluate our approach on four types of language understanding tasks – natural language inference, question answering, semantic similarity, and text classification. Our general task-agnostic model outperforms discriminatively trained models that employ architectures specifically crafted for each task, significantly improving upon the state of the art in 9 out of the 12 tasks studied. For instance, we achieve absolute improvements of 8.9% on commonsense reasoning (Stories Cloze Test) [40], 5.7% on question answering (RACE) [30], 1.5% on textual entailment (MultiNLI) [66] and 5.5% on the recently introduced GLUE multi-task benchmark [64]. We also analyzed zero-shot behaviors of the pre-trained model on four different settings and demonstrate that it acquires useful linguistic knowledge for downstream tasks.

## 2 Related Work

**Semi-supervised learning for NLP** Our work broadly falls under the category of semi-supervised learning for natural language. This paradigm has attracted significant interest, with applications to tasks like sequence labeling [24, 33, 57] or text classification [41, 70]. The earliest approaches used unlabeled data to compute word-level or phrase-level statistics, which were then used as features in a supervised model [33]. Over the last few years, researchers have demonstrated the benefits of using word embeddings [11, 39, 42], which are trained on unlabeled corpora, to improve performance on a variety of tasks [8, 11, 26, 45]. These approaches, however, mainly transfer word-level information, whereas we aim to capture higher-level semantics.

Recent approaches have investigated learning and utilizing more than word-level semantics from unlabeled data. Phrase-level or sentence-level embeddings, which can be trained using an unlabeled corpus, have been used to encode text into suitable vector representations for various target tasks [28, 32, 1, 36, 22, 12, 56, 31].

**Unsupervised pre-training** Unsupervised pre-training is a special case of semi-supervised learning where the goal is to find a good initialization point instead of modifying the supervised learning objective. Early works explored the use of the technique in image classification [20, 49, 63] and regression tasks [3]. Subsequent research [15] demonstrated that pre-training acts as a regularization scheme, enabling better generalization in deep neural networks. In recent work, the method has been used to help train deep neural networks on various tasks like image classification [69], speech recognition [68], entity disambiguation [17] and machine translation [48].

The closest line of work to ours involves pre-training a neural network using a language modeling objective and then fine-tuning it on a target task with supervision. Dai et al. [13] and Howard and Ruder [21] follow this method to improve text classification. However, although the pre-training phase helps capture some linguistic information, their usage of LSTM models restricts their prediction ability to a short range. In contrast, our choice of transformer networks allows us to capture longer-range linguistic structure, as demonstrated in our experiments. Further, we also demonstrate the effectiveness of our model on a wider range of tasks including natural language inference, paraphrase detection and story completion. Other approaches [43, 44, 38] use hidden representations from a

pre-trained language or machine translation model as auxiliary features while training a supervised model on the target task. This involves a substantial amount of new parameters for each separate target task, whereas we require minimal changes to our model architecture during transfer.

**Auxiliary training objectives** Adding auxiliary unsupervised training objectives is an alternative form of semi-supervised learning. Early work by Collobert and Weston [10] used a wide variety of auxiliary NLP tasks such as POS tagging, chunking, named entity recognition, and language modeling to improve semantic role labeling. More recently, Rei [50] added an auxiliary language modeling objective to their target task objective and demonstrated performance gains on sequence labeling tasks. Our experiments also use an auxiliary objective, but as we show, unsupervised pre-training already learns several linguistic aspects relevant to target tasks.

# 3 Framework

Our training procedure consists of two stages. The first stage is learning a high-capacity language model on a large corpus of text. This is followed by a fine-tuning stage, where we adapt the model to a discriminative task with labeled data.

## 3.1 Unsupervised pre-training

Given an unsupervised corpus of tokens $\mathcal{U} = \{u_1, \ldots, u_n\}$, we use a standard language modeling objective to maximize the following likelihood:

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta) \tag{1}$$

where $k$ is the size of the context window, and the conditional probability $P$ is modeled using a neural network with parameters $\Theta$. These parameters are trained using stochastic gradient descent [51].

In our experiments, we use a multi-layer *Transformer decoder* [34] for the language model, which is a variant of the transformer [62]. This model applies a multi-headed self-attention operation over the input context tokens followed by position-wise feedforward layers to produce an output distribution over target tokens:

$$\begin{aligned} h_0 &= UW_e + W_p \\ h_l &= \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n] \\ P(u) &= \texttt{softmax}(h_n W_e^T) \end{aligned} \tag{2}$$

where $U = (u_{-k}, \ldots, u_{-1})$ is the context vector of tokens, $n$ is the number of layers, $W_e$ is the token embedding matrix, and $W_p$ is the position embedding matrix.

## 3.2 Supervised fine-tuning

After training the model with the objective in Eq. 1, we adapt the parameters to the supervised target task. We assume a labeled dataset $\mathcal{C}$, where each instance consists of a sequence of input tokens, $x^1, \ldots, x^m$, along with a label $y$. The inputs are passed through our pre-trained model to obtain the final transformer block's activation $h_l^m$, which is then fed into an added linear output layer with parameters $W_y$ to predict $y$:

$$P(y | x^1, \ldots, x^m) = \texttt{softmax}(h_l^m W_y). \tag{3}$$

This gives us the following objective to maximize:

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y | x^1, \ldots, x^m). \tag{4}$$

We additionally found that including language modeling as an auxiliary objective to the fine-tuning helped learning by (a) improving generalization of the supervised model, and (b) accelerating convergence. This is in line with prior work [50, 43], who also observed improved performance with such an auxiliary objective. Specifically, we optimize the following objective (with weight $\lambda$):

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C}) \tag{5}$$

Overall, the only extra parameters we require during fine-tuning are $W_y$, and embeddings for delimiter tokens (described below in Section 3.3).

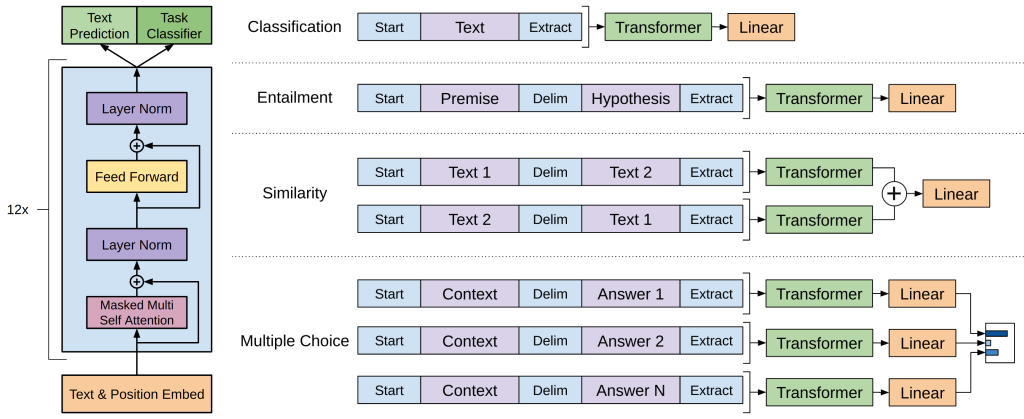


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

### 3.3 Task-specific input transformations

For some tasks, like text classification, we can directly fine-tune our model as described above. Certain other tasks, like question answering or textual entailment, have structured inputs such as ordered sentence pairs, or triplets of document, question, and answers. Since our pre-trained model was trained on contiguous sequences of text, we require some modifications to apply it to these tasks. Previous work proposed learning task specific architectures on top of transferred representations [44]. Such an approach re-introduces a significant amount of task-specific customization and does not use transfer learning for these additional architectural components. Instead, we use a traversal-style approach [52], where we convert structured inputs into an ordered sequence that our pre-trained model can process. These input transformations allow us to avoid making extensive changes to the architecture across tasks. We provide a brief description of these input transformations below and Figure 1 provides a visual illustration. All transformations include adding randomly initialized start and end tokens ($\langle s \rangle$, $\langle e \rangle$).

**Textual entailment** For entailment tasks, we concatenate the premise $p$ and hypothesis $h$ token sequences, with a delimiter token ($) in between.

**Similarity** For similarity tasks, there is no inherent ordering of the two sentences being compared. To reflect this, we modify the input sequence to contain both possible sentence orderings (with a delimiter in between) and process each independently to produce two sequence representations $h_l^m$ which are added element-wise before being fed into the linear output layer.

**Question Answering and Commonsense Reasoning** For these tasks, we are given a context document $z$, a question $q$, and a set of possible answers $\{a_k\}$. We concatenate the document context and question with each possible answer, adding a delimiter token in between to get $[z; q; \$; a_k]$. Each of these sequences are processed independently with our model and then normalized via a softmax layer to produce an output distribution over possible answers.

## 4 Experiments

### 4.1 Setup

**Unsupervised pre-training** We use the BooksCorpus dataset [71] for training the language model. It contains over 7,000 unique unpublished books from a variety of genres including Adventure, Fantasy, and Romance. Crucially, it contains long stretches of contiguous text, which allows the generative model to learn to condition on long-range information. An alternative dataset, the 1B Word Benchmark, which is used by a similar approach, ELMo [44], is approximately the same size

Table 1: A list of the different tasks and datasets used in our experiments.

| Task | Datasets |
|---|---|
| Natural language inference | SNLI [5], MultiNLI [66], Question NLI [64], RTE [4], SciTail [25] |
| Question Answering | RACE [30], Story Cloze [40] |
| Sentence similarity | MSR Paraphrase Corpus [14], Quora Question Pairs [9], STS Benchmark [6] |
| Classification | Stanford Sentiment Treebank-2 [54], CoLA [65] |

but is shuffled at a sentence level - destroying long-range structure. Our language model achieves a very low token level perplexity of 18.4 on this corpus.

**Model specifications** Our model largely follows the original transformer work [62]. We trained a 12-layer decoder-only transformer with masked self-attention heads (768 dimensional states and 12 attention heads). For the position-wise feed-forward networks, we used 3072 dimensional inner states. We used the Adam optimization scheme [27] with a max learning rate of 2.5e-4. The learning rate was increased linearly from zero over the first 2000 updates and annealed to 0 using a cosine schedule. We train for 100 epochs on minibatches of 64 randomly sampled, contiguous sequences of 512 tokens. Since layernorm [2] is used extensively throughout the model, a simple weight initialization of $N(0, 0.02)$ was sufficient. We used a bytepair encoding (BPE) vocabulary with 40,000 merges [53] and residual, embedding, and attention dropouts with a rate of 0.1 for regularization. We also employed a modified version of L2 regularization proposed in [37], with $w = 0.01$ on all non bias or gain weights. For the activation function, we used the Gaussian Error Linear Unit (GELU) [18]. We used learned position embeddings instead of the sinusoidal version proposed in the original work. We use the *ftfy* library[2] to clean the raw text in BooksCorpus, standardize some punctuation and whitespace, and use the *spaCy* tokenizer.[3]

**Fine-tuning details** Unless specified, we reuse the hyperparameter settings from unsupervised pre-training. We add dropout to the classifier with a rate of 0.1. For most tasks, we use a learning rate of 6.25e-5 and a batchsize of 32. Our model finetunes quickly and 3 epochs of training was sufficient for most cases. We use a linear learning rate decay schedule with warmup over 0.2% of training. $\lambda$ was set to 0.5.

### 4.2 Supervised fine-tuning

We perform experiments on a variety of supervised tasks including natural language inference, question answering, semantic similarity, and text classification. Some of these tasks are available as part of the recently released GLUE multi-task benchmark [64], which we make use of. Figure 1 provides an overview of all the tasks and datasets.

**Natural Language Inference** The task of natural language inference (NLI), also known as recognizing textual entailment, involves reading a pair of sentences and judging the relationship between them from one of `entailment`, `contradiction` or `neutral`. Although there has been a lot of recent interest [58, 35, 44], the task remains challenging due to the presence of a wide variety of phenomena like lexical entailment, coreference, and lexical and syntactic ambiguity. We evaluate on five datasets with diverse sources, including image captions (SNLI), transcribed speech, popular fiction, and government reports (MNLI), Wikipedia articles (QNLI), science exams (SciTail) or news articles (RTE).

Table 2 details various results on the different NLI tasks for our model and previous state-of-the-art approaches. Our method significantly outperforms the baselines on four of the five datasets, achieving absolute improvements of upto 1.5% on MNLI, 5% on SciTail, 5.8% on QNLI and 0.6% on SNLI over the previous best results. This demonstrates our model's ability to better reason over multiple sentences, and handle aspects of linguistic ambiguity. On RTE, one of the smaller datasets we evaluate on (2490 examples), we achieve an accuracy of 56%, which is below the 61.7% reported by a multi-task biLSTM model. Given the strong performance of our approach on larger NLI datasets, it is likely our model will benefit from multi-task training as well but we have not explored this currently.

[2] `https://ftfy.readthedocs.io/en/latest/`

[3] `https://spacy.io/`

Table 2: Experimental results on natural language inference tasks, comparing our model with current state-of-the-art methods. 5x indicates an ensemble of 5 models. All datasets use accuracy as the evaluation metric.

| Method | MNLI-m | MNLI-mm | SNLI | SciTail | QNLI | RTE |
|---|---|---|---|---|---|---|
| ESIM + ELMo [44] (5x) | - | - | 89.3 | - | - | - |
| CAFE [58] (5x) | 80.2 | 79.0 | 89.3 | - | - | - |
| Stochastic Answer Network [35] (3x) | 80.6 | 80.1 | - | - | - | - |
| CAFE [58] | 78.7 | 77.9 | 88.5 | 83.3 | | |
| GenSen [64] | 71.4 | 71.3 | - | - | 82.3 | 59.2 |
| Multi-task BiLSTM + Attn [64] | 72.2 | 72.1 | - | - | 82.1 | **61.7** |
| Finetuned Transformer LM (ours) | **82.1** | **81.4** | **89.9** | **88.3** | **88.1** | 56.0 |

Table 3: Results on question answering and commonsense reasoning, comparing our model with current state-of-the-art methods.. 9x means an ensemble of 9 models.

| Method | Story Cloze | RACE-m | RACE-h | RACE |
|---|---|---|---|---|
| val-LS-skip [55] | 76.5 | - | - | - |
| Hidden Coherence Model [7] | 77.6 | - | - | - |
| Dynamic Fusion Net [67] (9x) | - | 55.6 | 49.4 | 51.2 |
| BiAttention MRU [59] (9x) | - | 60.2 | 50.3 | 53.3 |
| Finetuned Transformer LM (ours) | **86.5** | **62.9** | **57.4** | **59.0** |

**Question answering and commonsense reasoning** Another task that requires aspects of single and multi-sentence reasoning is question answering. We use the recently released RACE dataset [30], consisting of English passages with associated questions from middle and high school exams. This corpus has been shown to contain more reasoning type questions that other datasets like CNN [19] or SQuaD [47], providing the perfect evaluation for our model which is trained to handle long-range contexts. In addition, we evaluate on the Story Cloze Test [40], which involves selecting the correct ending to multi-sentence stories from two options. On these tasks, our model again outperforms the previous best results by significant margins - up to 8.9% on Story Cloze, and 5.7% overall on RACE. This demonstrates the ability of our model to handle long-range contexts effectively.

**Semantic Similarity** Semantic similarity (or paraphrase detection) tasks involve predicting whether two sentences are semantically equivalent or not. The challenges lie in recognizing rephrasing of concepts, understanding negation, and handling syntactic ambiguity. We use three datasets for this task – the Microsoft Paraphrase corpus (MRPC) [14] (collected from news sources), the Quora Question Pairs (QQP) dataset [9], and the Semantic Textual Similarity benchmark (STS-B) [6]. We obtain state-of-the-art results on two of the three semantic similarity tasks (Table 4) with a 1 point absolute gain on STS-B. The performance delta on QQP is significant, with a 4.2% absolute improvement over Single-task BiLSTM + ELMo + Attn.

**Classification** Finally, we also evaluate on two different text classification tasks. The Corpus of Linguistic Acceptability (CoLA) [65] contains expert judgements on whether a sentence is grammatical or not, and tests the innate linguistic bias of trained models. The Stanford Sentiment Treebank (SST-2) [54], on the other hand, is a standard binary classification task. Our model obtains an score of 45.4 on CoLA, which is an especially big jump over the previous best result of 35.0, showcasing the innate linguistic bias learned by our model. The model also achieves 91.3% accuracy on SST-2, which is competitive with the state-of-the-art results. We also achieve an overall score of 72.8 on the GLUE benchmark, which is significantly better than the previous best of 68.9.

Table 4: Semantic similarity and classification results, comparing our model with current state-of-the-art methods. All task evaluations in this table were done using the GLUE benchmark. (*mc*= Mathews correlation, *acc*=Accuracy, *pc*=Pearson correlation)

| **Method** | **Classification** | | **Semantic Similarity** | | | **GLUE** |
|---|---|---|---|---|---|---|
| | CoLA (mc) | SST2 (acc) | MRPC (F1) | STSB (pc) | QQP (F1) | |
| Sparse byte mLSTM [16] | - | **93.2** | - | - | - | - |
| TF-KLD [23] | - | - | **86.0** | - | - | - |
| ECNU (mixed ensemble) [60] | - | - | - | 81.0 | - | - |
| Single-task BiLSTM + ELMo + Attn [64] | 35.0 | 90.2 | 80.2 | 55.5 | 66.1 | 64.8 |
| Multi-task BiLSTM + ELMo + Attn [64] | 18.9 | 91.6 | 83.5 | 72.8 | 63.3 | 68.9 |
| Finetuned Transformer LM (ours) | **45.4** | 91.3 | 82.3 | **82.0** | **70.3** | **72.8** |

Overall, our approach achieves new state-of-the-art results in 9 out of the 12 datasets we evaluate on, outperforming ensembles in many cases. Our results also indicate that our approach works well across datasets of different sizes, from smaller datasets such as STS-B (≈5.7k training examples) – to the largest one – SNLI (≈550k training examples).

## 5 Analysis

**Impact of number of layers transferred** We observed the impact of transferring a variable number of layers from unsupervised pre-training to the supervised target task. Figure 2(left) illustrates the performance of our approach on MultiNLI and RACE as a function of the number of layers transferred. We observe the standard result that transferring embeddings improves performance and that each transformer layer provides further benefits up to 9% for full transfer on MultiNLI. This indicates that each layer in the pre-trained model contains useful functionality for solving target tasks.

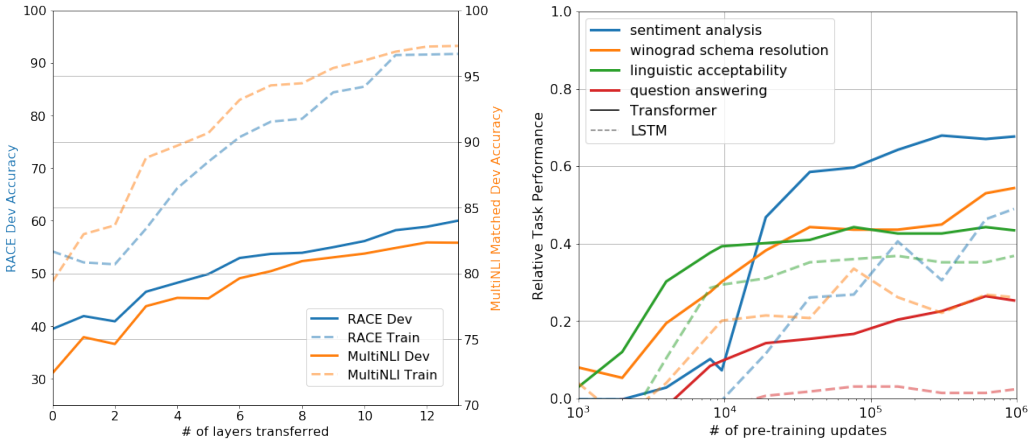


Figure 2: (**left**) Effect of transferring increasing number of layers from the pre-trained language model on RACE and MultiNLI. (**right**) Plot showing the evolution of zero-shot performance on different tasks as a function of LM pre-training updates. Performance per task is normalized between a random guess baseline and the current state-of-the-art with a single model.

**Zero-shot Behaviors** We'd like to better understand why language model pre-training of transformers is effective. A hypothesis is that the underlying generative model learns to perform many of the tasks we evaluate on in order to improve its language modeling capability and that the more structured

Table 5: Analysis of various model ablations on different tasks. Avg. score is a unweighted average of all the results. (*mc*= Mathews correlation, *acc*=Accuracy, *pc*=Pearson correlation)

| Method | Avg. Score | CoLA (mc) | SST2 (acc) | MRPC (F1) | STSB (pc) | QQP (F1) | MNLI (acc) | QNLI (acc) | RTE (acc) |
|---|---|---|---|---|---|---|---|---|---|
| Transformer w/ aux LM (full) | 74.7 | 45.4 | 91.3 | 82.3 | 82.0 | **70.3** | **81.8** | **88.1** | **56.0** |
| Transformer w/o pre-training | 59.9 | 18.9 | 84.0 | 79.4 | 30.9 | 65.5 | 75.7 | 71.2 | 53.8 |
| Transformer w/o aux LM | **75.0** | **47.9** | **92.0** | **84.9** | **83.2** | 69.8 | 81.1 | 86.9 | 54.4 |
| LSTM w/ aux LM | 69.1 | 30.3 | 90.5 | 83.2 | 71.8 | 68.1 | 73.7 | 81.1 | 54.6 |

attentional memory of the transformer assists in transfer compared to LSTMs. We designed a series of heuristic solutions that use the underlying generative model to perform tasks without supervised finetuning. We visualize the effectiveness of these heuristic solutions over the course of generative pre-training in Fig 2(right). We observe the performance of these heuristics is stable and steadily increases over training suggesting that generative pretraining supports the learning of a wide variety of task relevant functionality. We also observe the LSTM exhibits higher variance in its zero-shot performance suggesting that the inductive bias of the Transformer architecture assists in transfer.

For CoLA (linguistic acceptability), examples are scored as the average token log-probability the generative model assigns and predictions are made by thresholding. For SST-2 (sentiment analysis), we append the token *very* to each example and restrict the language model's output distribution to only the words *positive* and *negative* and guess the token it assigns higher probability to as the prediction. For RACE (question answering), we pick the answer the generative model assigns the highest average token log-probability when conditioned on the document and question. For DPRD [46] (winograd schemas), we replace the definite pronoun with the two possible referrents and predict the resolution that the generative model assigns higher average token log-probability to the rest of the sequence after the substitution.

**Ablation studies** We perform three different ablation studies (Table 5). First, we examine the performance of our method without the auxiliary LM objective during fine-tuning. We observe that the auxiliary objective helps on the NLI tasks and QQP. Overall, the trend suggests that larger datasets benefit from the auxiliary objective but smaller datasets do not. Second, we analyze the effect of the Transformer by comparing it with a single layer 2048 unit LSTM using the same framework. We observe a 5.6 average score drop when using the LSTM instead of the Transformer. The LSTM only outperforms the Transformer on one dataset – MRPC. Finally, we also compare with our transformer architecture directly trained on supervised target tasks, without pre-training. We observe that the lack of pre-training hurts performance across all the tasks, resulting in a 14.8% decrease compared to our full model.

## 6 Conclusion

We introduced a framework for achieving strong natural language understanding with a single task-agnostic model through generative pre-training and discriminative fine-tuning. By pre-training on a diverse corpus with long stretches of contiguous text our model acquires significant world knowledge and ability to process long-range dependencies which are then successfully transferred to solving discriminative tasks such as question answering, semantic similarity assessment, entailment determination, and text classification, improving the state of the art on 9 of the 12 datasets we study. Using unsupervised (pre-)training to boost performance on discriminative tasks has long been an important goal of Machine Learning research. Our work suggests that achieving significant performance gains is indeed possible, and offers hints as to what models (Transformers) and data sets (text with long range dependencies) work best with this approach. We hope that this will help enable new research into unsupervised learning, for both natural language understanding and other domains, further improving our understanding of how and when unsupervised learning works.

## References

[1] S. Arora, Y. Liang, and T. Ma. A simple but tough-to-beat baseline for sentence embeddings. 2016.

[2] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160, 2007.

[4] L. Bentivogli, P. Clark, I. Dagan, and D. Giampiccolo. The fifth pascal recognizing textual entailment challenge. In *TAC*, 2009.

[5] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. *EMNLP*, 2015.

[6] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

[7] S. Chaturvedi, H. Peng, and D. Roth. Story comprehension for predicting what happens next. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1603–1614, 2017.

[8] D. Chen and C. Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

[9] Z. Chen, H. Zhang, X. Zhang, and L. Zhao. Quora question pairs. https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs, 2018.

[10] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[11] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[12] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. *EMNLP*, 2017.

[13] A. M. Dai and Q. V. Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3079–3087, 2015.

[14] W. B. Dolan and C. Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[15] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

[16] S. Gray, A. Radford, and K. P. Diederik. Gpu kernels for block-sparse weights. 2017.

[17] Z. He, S. Liu, M. Li, M. Zhou, L. Zhang, and H. Wang. Learning entity representation for entity disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 30–34, 2013.

[18] D. Hendrycks and K. Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *arXiv preprint arXiv:1606.08415*, 2016.

[19] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701, 2015.

[20] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[21] J. Howard and S. Ruder. Universal language model fine-tuning for text classification. *Association for Computational Linguistics (ACL)*, 2018.

[22] Y. Jernite, S. R. Bowman, and D. Sontag. Discourse-based objectives for fast unsupervised sentence representation learning. *arXiv preprint arXiv:1705.00557*, 2017.

[23] Y. Ji and J. Eisenstein. Discriminative improvements to distributional sentence similarity. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 891–896, 2013.

[24] F. Jiao, S. Wang, C.-H. Lee, R. Greiner, and D. Schuurmans. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 209–216. Association for Computational Linguistics, 2006.

[25] T. Khot, A. Sabharwal, and P. Clark. Scitail: A textual entailment dataset from science question answering. In *Proceedings of AAAI*, 2018.

[26] Y. Kim. Convolutional neural networks for sentence classification. *EMNLP*, 2014.

[27] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[28] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

[29] N. Kitaev and D. Klein. Constituency parsing with a self-attentive encoder. *ACL*, 2018.

[30] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy. Race: Large-scale reading comprehension dataset from examinations. *EMNLP*, 2017.

[31] G. Lample, L. Denoyer, and M. Ranzato. Unsupervised machine translation using monolingual corpora only. *ICLR*, 2018.

[32] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

[33] P. Liang. *Semi-supervised learning for natural language*. PhD thesis, Massachusetts Institute of Technology, 2005.

[34] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer. Generating wikipedia by summarizing long sequences. *ICLR*, 2018.

[35] X. Liu, K. Duh, and J. Gao. Stochastic answer networks for natural language inference. *arXiv preprint arXiv:1804.07888*, 2018.

[36] L. Logeswaran and H. Lee. An efficient framework for learning sentence representations. *ICLR*, 2018.

[37] I. Loshchilov and F. Hutter. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 2017.

[38] B. McCann, J. Bradbury, C. Xiong, and R. Socher. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6297–6308, 2017.

[39] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[40] N. Mostafazadeh, M. Roth, A. Louis, N. Chambers, and J. Allen. Lsdsem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 46–51, 2017.

[41] K. Nigam, A. McCallum, and T. Mitchell. Semi-supervised text classification using em. *Semi-Supervised Learning*, pages 33–56, 2006.

[42] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[43] M. E. Peters, W. Ammar, C. Bhagavatula, and R. Power. Semi-supervised sequence tagging with bidirectional language models. *ACL*, 2017.

[44] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *NAACL*, 2018.

[45] Y. Qi, D. S. Sachan, M. Felix, S. J. Padmanabhan, and G. Neubig. When and why are pre-trained word embeddings useful for neural machine translation? *NAACL*, 2018.

[46] A. Rahman and V. Ng. Resolving complex cases of definite pronouns: the winograd schema challenge. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 777–789. Association for Computational Linguistics, 2012.

[47] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension of text. *EMNLP*, 2016.

[48] P. Ramachandran, P. J. Liu, and Q. V. Le. Unsupervised pretraining for sequence to sequence learning. *arXiv preprint arXiv:1611.02683*, 2016.

[49] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, pages 1137–1144, 2007.

[50] M. Rei. Semi-supervised multitask learning for sequence labeling. *ACL*, 2017.

[51] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[52] T. Rocktäschel, E. Grefenstette, K. M. Hermann, T. Kočiskỳ, and P. Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

[53] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[54] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. 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, 2013.

[55] S. Srinivasan, R. Arora, and M. Riedl. A simple and effective approach to the story cloze test. *arXiv preprint arXiv:1803.05547*, 2018.

[56] S. Subramanian, A. Trischler, Y. Bengio, and C. J. Pal. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*, 2018.

[57] J. Suzuki and H. Isozaki. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. *Proceedings of ACL-08: HLT*, pages 665–673, 2008.

[58] Y. Tay, L. A. Tuan, and S. C. Hui. A compare-propagate architecture with alignment factorization for natural language inference. *arXiv preprint arXiv:1801.00102*, 2017.

[59] Y. Tay, L. A. Tuan, and S. C. Hui. Multi-range reasoning for machine comprehension. *arXiv preprint arXiv:1803.09074*, 2018.

[60] J. Tian, Z. Zhou, M. Lan, and Y. Wu. Ecnu at semeval-2017 task 1: Leverage kernel-based traditional nlp features and neural networks to build a universal model for multilingual and cross-lingual semantic textual similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 191–197, 2017.

[61] Y. Tsvetkov. Opportunities and challenges in working with low-resource languages. *CMU*, 2017.

[62] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010, 2017.

[63] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[64] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[65] A. Warstadt, A. Singh, and S. R. Bowman. Corpus of linguistic acceptability. http://nyu-mll.github.io/cola, 2018.

[66] A. Williams, N. Nangia, and S. R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *NAACL*, 2018.

[67] Y. Xu, J. Liu, J. Gao, Y. Shen, and X. Liu. Towards human-level machine reading comprehension: Reasoning and inference with multiple strategies. *arXiv preprint arXiv:1711.04964*, 2017.

[68] D. Yu, L. Deng, and G. Dahl. Roles of pre-training and fine-tuning in context-dependent dbn-hmms for real-world speech recognition. In *Proc. NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[69] R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, volume 1, page 6, 2017.

[70] X. Zhu. Semi-supervised learning literature survey. 2005.

[71] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.

# GENERATING WIKIPEDIA BY SUMMARIZING LONG SEQUENCES

**Peter J. Liu***, **Mohammad Saleh***,
**Etienne Pot**†, **Ben Goodrich, Ryan Sepassi, Łukasz Kaiser, Noam Shazeer**
Google Brain
Mountain View, CA
{peterjliu,msaleh,epot,bgoodrich,rsepassi,lukaszkaiser,noam}@google.com



## ABSTRACT

We show that generating English Wikipedia articles can be approached as a multi-document summarization of source documents. We use extractive summarization to coarsely identify salient information and a neural abstractive model to generate the article. For the abstractive model, we introduce a decoder-only architecture that can scalably attend to very long sequences, much longer than typical encoder-decoder architectures used in sequence transduction. We show that this model can generate fluent, coherent multi-sentence paragraphs and even whole Wikipedia articles. When given reference documents, we show it can extract relevant factual information as reflected in perplexity, ROUGE scores and human evaluations.

## 1 INTRODUCTION

The sequence-to-sequence framework has demonstrated success in natural-language sequence transduction tasks such as machine translation. More recently, neural techniques have been applied to do single-document, abstractive (paraphrasing) text summarization of news articles (Rush et al. (2015), Nallapati et al. (2016)). In this prior work, the input to supervised models ranged from the first sentence to the entire text of an article, and they are trained end-to-end to predict reference summaries. Doing this end-to-end requires a significant number of parallel article-summary pairs since language understanding is a pre-requisite to generate fluent summaries.

In contrast, we consider the task of multi-document summarization, where the input is a collection of related documents from which a summary is distilled. Prior work has focused on extractive summarization, which select sentences or phrases from the input to form the summaries, rather than generating new text. There has been limited application of abstractive neural methods and one possible reason is the paucity of large, labeled datasets.

In this work, we consider English Wikipedia as a supervised machine learning task for multi-document summarization where the input is comprised of a Wikipedia topic (title of article) and a collection of non-Wikipedia reference documents, and the target is the Wikipedia article text. We describe the first attempt to abstractively generate the first section, or *lead*, of Wikipedia articles conditioned on reference text. In addition to running strong baseline models on the task, we modify the Transformer architecture (Vaswani et al., 2017) to only consist of a decoder, which performs better in the case of longer input sequences compared to recurrent neural network (RNN) and Transformer encoder-decoder models. Finally we show our modeling improvements allow us to generate entire Wikipedia articles.

*Joint first-authors. Ordered randomly.
†Work done as a member of the Google Brain Residency (g.co/brainresidency)

## 2 RELATED WORK

### 2.1 OTHER DATASETS USED IN NEURAL ABSTRACTIVE SUMMARIZATION

Neural abstractive summarization was pioneered in Rush et al. (2015), where they train headline generation models using the English Gigaword corpus (Graff & Cieri, 2003), consisting of news articles from number of publishers. However, the task is more akin to sentence paraphrasing than summarization as only the first sentence of an article is used to predict the headline, another sentence. RNN-based encoder-decoder models with attention (seq2seq) perform very well on this task in both ROUGE (Lin, 2004), an automatic metric often used in summarization, and human evaluation (Chopra et al., 2016).

In Nallapati et al. (2016), an abstractive summarization dataset is proposed by modifying a question-answering dataset of news articles paired with story highlights from Daily Mail and CNN. This task is more difficult than headline-generation because the information used in the highlights may come from many parts of the article and not only the first sentence. One downside of the dataset is that it has an order-of-magnitude fewer parallel examples (310k vs. 3.8M) to learn from. Standard seq2seq models with attention do less well, and a number of techniques are used to augment performance. Another downside is that it is unclear what the guidelines are for creating story highlights and it is obvious that there are significant stylistic differences between the two news publishers.

In our work we also train neural abstractive models, but in the multi-document regime with Wikipedia. As can be seen in Table 1, the input and output text are generally much larger, with significant variance depending on the article. The summaries (Wikipedia lead) are multiple sentences and sometimes multiple paragraphs, written in a fairly uniform style as encouraged by the Wikipedia Manual of Style[1]. However, the input documents may consist of documents of arbitrary style originating from arbitrary sources.

We also show in Table 1 the ROUGE-1 recall scores of the output given the input, which is the proportion of unigrams/words in the output co-occuring in the input. A higher score corresponds to a dataset more amenable to extractive summarization. In particular, if the output is completely embedded somewhere in the input (e.g. a wiki-clone), the score would be 100. Given a score of only 59.2 compared to 76.1 and 78.7 for other summarization datasets shows that ours is the least amenable to purely extractive methods.

### 2.2 TASKS INVOLVING WIKIPEDIA

There is a rich body of work incorporating Wikipedia for machine learning tasks, including question-answering (Hewlett et al. (2016), Rajpurkar et al. (2016)) and information extraction (Lehmann et al., 2015), and text generation from structured data (Lebret et al., 2016).

The closest work to ours involving generating Wikipedia is Sauper & Barzilay (2009), where articles are generated extractively (instead of abstractively in our case) from reference documents using learned templates. The Wikipedia articles are restricted to two categories, whereas we use all article types. The reference documents are obtained from a search engine, with the Wikipedia topic used as query similar to our search engine references. However we also show results with documents only found in the References section of the Wikipedia articles.

### 2.3 TRANSFORMER MODELS

Previous work on neural abstractive summarization relies on RNNs as fundamental modules, mirroring techniques successful in machine translation (MT). Recently, state-of-the-art MT results were obtained using a non-recurrent architecture, called the Transformer (Vaswani et al., 2017). The lack of recurrence enables greater within-training-example parallelization, at the cost of quadratic complexity in the input sequence length. We find the Transformer transfers well to medium length, input sequence summarization and describe modifications to better handle longer sequences.

[1] https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style

Table 1: Order of magnitude input/output sizes and unigram recall for summarization datasets.

| Dataset | Input | Output | # examples | ROUGE-1 R |
|---|---|---|---|---|
| Gigaword (Graff & Cieri, 2003) | $10^1$ | $10^1$ | $10^6$ | 78.7 |
| CNN/DailyMail (Nallapati et al., 2016) | $10^2$–$10^3$ | $10^1$ | $10^5$ | 76.1 |
| WikiSum (ours) | $10^2$–$10^6$ | $10^1$–$10^3$ | $10^6$ | 59.2 |

Table 2: Percentiles for different aspects of WikiSum dataset. Size is in number of words.

| Percentile | 20 | 40 | 50 | 60 | 80 | 100 |
|---|---|---|---|---|---|---|
| Lead Size | 37 | 62 | 78 | 98 | 166 | 10,034 |
| Num Citations | 1 | 2 | 2 | 3 | 5 | 1,029 |
| Citations Size | 562 | 1,467 | 2,296 | 3,592 | 10,320 | 6,159,463 |
| Num Search Results | 10 | 20 | 26 | 31 | 46 | 2,095 |
| Search Results Size | 1,1691 | 33,989 | 49,222 | 68,681 | 135,533 | 5,355,671 |

## 3 English Wikipedia as a multi-document summarization dataset

Wikipedia, being an encyclopedia, can be viewed as a collection of summaries on various topics given by their title, e.g. "Canada" or "Machine Learning". The source material to be summarized can be viewed as all reputable documents on the Web or books; however, to make the problem more tractable we consider the following subsets of all documents, $D$:

1. Cited sources: A Wikipedia article that conforms to the style guidelines should be well-supported by citations found in the *References* section of Wikipedia articles. For each article, $a_i$, we extract all text without markup from crawlable citation documents, $C_i \subset D$, to use as input to our method.
2. Web Search results: To expand the collection of reference documents, we crawl the search results from the Google search engine, using the article section titles as queries. For each query, we collect 10 result pages. From this collection we remove the Wikipedia article itself, which is often among the top results. We also remove "clones", which are detected when there is a high-level of unigram overlap with the article (details provided in A.2.1). We denote these refined search results for an article, $a_i$, as $S_i \subset D$. Similar to $C_i$, we extract only the text to use as input.

Table 2 describes overall properties of our WikiSum dataset. Many articles have few citations, motivating our supplementation of the source documents with web search results. On the other hand, citations when available, tend to be of higher-quality. When counting the total words in the entire dataset, it is orders-of-magnitude larger than previous summarization datasets.

To have consistent train/development/test data across corpus-comparison experiments, we restrict the articles to those with at least one crawlable citation. We divide the articles roughly into 80/10/10 for train/development/test subsets, resulting in 1865750, 233252, and 232998 examples respectively.

## 4 Methods and Models

Because the amount of text in input reference documents $(C_i, S_i)$ can be very large (see Table 2) it is infeasible to train an end-to-end abstractive model given the memory constraints of current hardware. Hence, we first coarsely select a subset of the input using extractive summarization. The second stage involves training an abstractive model that generates the Wikipedia text while conditioning on this extraction. This two-stage process is inspired by by how humans might summarize multiple long documents: First highlight pertinent information, then conditionally generate the summary based on the highlights.

### 4.1 EXTRACTIVE STAGE

We investigate three extractive methods from the summarization literature, along with a trivial and cheating method, to assess the importance of this stage. For each article, $a_i$ we create a ranked list of paragraphs, $\{p^i_{R_i(j)}\}$, occurring in $(C_i, S_i)$ where $R_i(j)$ is the rank of the $j$th paragraph $p^i_j$ of $(C_i, S_i)$. From this we select the first $L$ tokens as input to the second abstractive stage.

1. *Identity*: As a trivial baseline extractor, we simply use the first $L$ tokens of the input.
2. *tf-idf*: A non-trivial ranking is to consider ranking paragraphs as documents in a query-retrieval problem, where the query is the title of the article, $T(a_i)$. We compute tf-idf (Ramos et al., 2003) for the query, with respect to the documents, $\{p^i_j\}$. That is, we summate for each word in the query
$$N_w \cdot log(\frac{N_d}{N_{dw}})$$
where $N_w$, $N_d$, and $N_{dw}$ are the count of the word in the document, total number of documents, and total number of documents containing the word, respectively.
3. *TextRank* (Mihalcea & Tarau, 2004): A weighted graph is defined where text units are nodes and edges are defined by a similarity measure based on word overlap. An algorithm similar to PageRank (Page et al., 1999) is then used to compute the ranking of text units. We used paragraphs for the text units.
4. *SumBasic* (Nenkova & Vanderwende, 2005): Word frequencies in the input text are used to assign scores to words, which are in turn used to score sentences. After selecting the best scoring sentence, words in it have their scores reduced, and the process is repeated until the desired summary length is reached.
5. *Cheating* To further demonstrate the quality of extraction on the final performance, we implement a cheating extractor that ranks $\{p^i_j\}$ using recall of bigrams in the ground truth text:
$$d(p^i_j, a_i) = \frac{bigrams(p^i_j) \cap bigrams(a_i)}{bigrams(a_i)} \quad (1)$$

### 4.2 ABSTRACTIVE STAGE

#### 4.2.1 DATA REPRESENTATION

Given the ordered paragraphs $\{p^i_{R_i(j)}\}$, we derive the raw text input simply as the concatenation of the paragraphs in order, the most relevant at the beginning, and prefixed with the title.

We then encode the text using sub-word tokenization similar to Wu et al. (2016) with a vocabulary size of 32,000 yielding tokenized input, $x_i$:
$$text_i = T(a_i) \| \{p^i_{R_i(j)}\}$$
$$tokenize(text_i) = x_i = (x^1_i, x^2_i, ..., x^{n_i}_i)$$

For various values of $L$ in experiments, we truncate the tokens to form the input sequence:
$$m^L_i = (x^1_i, ...x^{min(L,n_i)}_i)$$

For the output, we use the same vocabulary and tokenization for the Wikipedia lead text but do not do any truncation across experiments.

Next we describe the abstractive models, $W$, that learn to write articles, $a_i = W(m^L_i)$, which we treat as a sequence transduction problem from very long input sequences (up to $L = 11000$) to medium output sequences (typically less than 500).

#### 4.2.2 BASELINE MODELS

As a baseline we apply the standard LSTM encoder-decoder with attention (seq2seq-att) as in Bahdanau et al. (2014) to this task. As is typical we train to optimize the maximum-likelihood objective:

$$y_i = tokenize(a_i)$$
$$\prod_{i=1}^{N} p(y_i|m_i^L)$$

A stronger, more recent baseline that we use is the non-recurrent Transformer model described in 2.3, which also has symmetric encoder and decoder modules (T-ED).

#### 4.2.3 TRANSFORMER DECODER (T-D)

We introduce a simple but effective modification to T-ED for long sequences that drops the encoder module (almost reducing model parameters by half for a given hyper-parameter set), combines the input and output sequences into a single "sentence" and is trained as a standard language model.

That is, we convert a sequence-transduction example $(m^1, ..., m^n) \mapsto (y^1, ..., y^\eta)$ into the sentence $(w^1, ..., w^{n+\eta+1}) = (m^1, ..., m^n, \delta, y^1, ..., y^\eta)$, where $\delta$ is a special separator token and train a model to predict the next word given the previous ones:

$$p(w^1, ..., w^{n+\eta}) = \prod_{j=1}^{n+\eta} p(w^i|w^1, ..., w^{j-1})$$

Since the model is forced to predict the next token in the input, $m$, as well as $y$, error signals are propagated from both input and output time-steps during training. We also suspect that for monolingual text-to-text tasks redundant information is re-learned about language in the encoder and decoder. We believe this allows for easier optimization and empirically observe this with longer sequences (see Section 5.3). Note that because of the self-attention of the Transformer, when generating the next token, attention from both $m$ and $y$ are considered. At inference we provide the input sequence, $m_i$, initially, and auto-regressively generate the output, $y_i$, as normal.

#### 4.2.4 TRANSFORMER DECODER WITH MEMORY-COMPRESSED ATTENTION (T-DMCA)

To re-use the terminology used to describe the Transformer, the attention is a function of a query ($Q$) and set of key ($K$) and value ($V$) pairs. To handle longer sequences, we modify the multi-head self-attention of the Transformer to reduce memory usage by limiting the dot products between $Q$ and $K$ in:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

**Local attention**: Sequence tokens are divided into blocks of similar length and attention is performed in each block independently. As the attention memory cost per block becomes constant, this modification allow us to keep the number of activations linear with respect to the sequence length. In our experiments, we choose to have blocks of 256 tokens.

**Memory-compressed attention**: After projecting the tokens into the query, key, and value embeddings, we reduce the number of keys and values by using a strided convolution. The number of queries remains unchanged. This modification allows us to divide the number of activations by a compression factor. In our experiments we use convolution kernels of size 3 with stride 3. In contrast to local attention layers, which only capture the local information within a block, the memory-compressed attention layers are able to exchange information globally on the entire sequence.

These modifications (see Figure 1) allow us in practice to process sequences 3x in length over the T-D model. For both local and memory-compressed attention, masking is added to prevent the queries from attending to future keys and values. Our final architecture is a 5-layer network (LMLML) alternating between local-attention (L) layers and memory-compressed attention (M) layers (in Vaswani et al. (2017) it is 6 identical layers). We also added in some experiments one mixture of experts (MoE) layer (Shazeer et al., 2017) to increase the network's capacity.

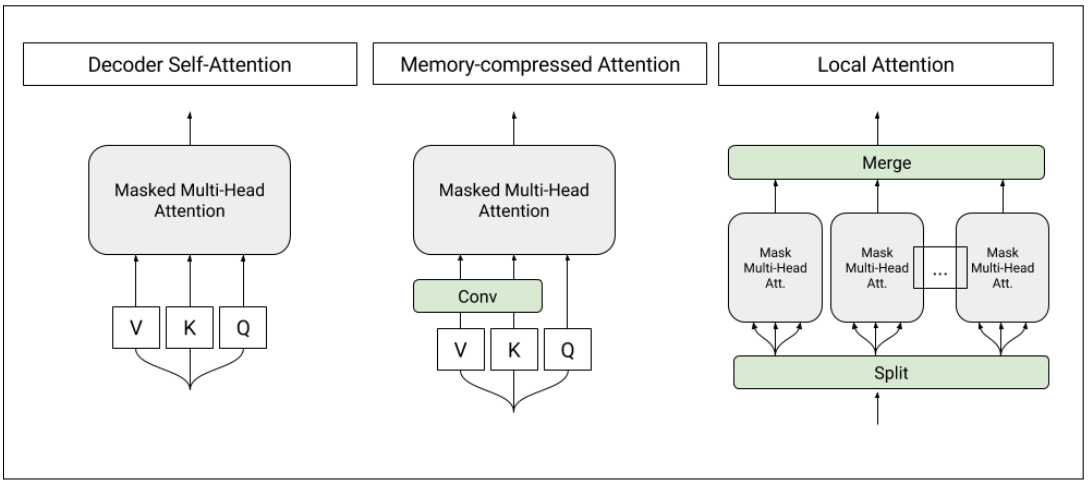


Figure 1: The architecture of the self-attention layers used in the T-DMCA model. Every attention layer takes a sequence of tokens as input and produces a sequence of similar length as the output. **Left:** Original self-attention as used in the transformer-decoder. **Middle:** Memory-compressed attention which reduce the number of keys/values. **Right:** Local attention which splits the sequence into individual smaller sub-sequences. The sub-sequences are then merged together to get the final output sequence.

# 5 EXPERIMENTS

## 5.1 EVALUATION

In experiments we evaluate based on perplexity (per-wordpiece), a common language modeling metric, and ROUGE-L F1 (version `ROUGE-1.5.5`), a common metric used in comparing candidate and reference summaries. Note the F1 flavor of ROUGE is more appropriate in this setting as we do not explicitly constrain the output length in abstractive models; it is the harmonic mean of ROUGE-Recall (which favors long summaries) and ROUGE-Precision (which favors short summaries).

Although optimizing ROUGE directly has been shown to not always yield the best summaries as evaluated by human judgment (Paulus et al., 2017), we found that for our task optimizing for perplexity correlates with increased ROUGE and human judgment. We suspect that the relatively uniform style of Wikipedia articles makes ROUGE more appropriate here than in general abstractive summarization tasks.

## 5.2 MODEL TRAINING DETAILS AND DECODING

For all abstractive model training, we use the open-source `tensor2tensor`[2] library.

The seq2seq baseline had a hidden size of 128 with 2 layers (we use the hyper-parameter set defined in the library as `lstm_attention`).

For the Transformer encoder-decoder (T-ED), we use the hyper-parameter set `transfomer_base_v1` and train for 1 million steps. Models exhibited very little overfitting and did not require early-stopping. The Transformer Decoder (T-D) was identical to the decoder part of T-ED. The T-DMCA model is similar to T-D, but with the enhancements described in section 4.2.4.

Unless otherwise stated, during decoding we use a beam search of size 4 and length penalty $\alpha = 0.6$ (Wu et al., 2016) and decode until an end-of-sequence token is reached.

[2] `https://github.com/tensorflow/tensor2tensor`

Table 3: Comparison of extractive method and corpus with $L = 500$, and the Transformer E-D model

| Extractor | Corpus | Test log-perplexity | ROUGE-L |
|---|---|---|---|
| *cheating* | combined | 1.72975 | 59.3 |
| *tf-idf* | combined | 2.46645 | 34.2 |
| *tf-idf* | citations-only | 3.04299 | 22.6 |
| *tf-idf* | search-only | 3.56593 | 2.8 |
| *identity* | combined | 4.80215 | 4.0 |

### 5.3 Results and Discussion

There are four main dimensions we vary in experiments in generating Wikipedia lead sections:

1. Extractive method: *SumBasic*, *TextRank*, *tf-idf*, *identity*, *cheating extractor*
2. Input corpus: *citations*, *search results*, *combined*
3. Abstractive model input length, $L$: We try values between 100 and 11000.
4. Abstractive model architecture: *seq2seq-att*, *T-ED*, *T-D*, *T-DMCA*

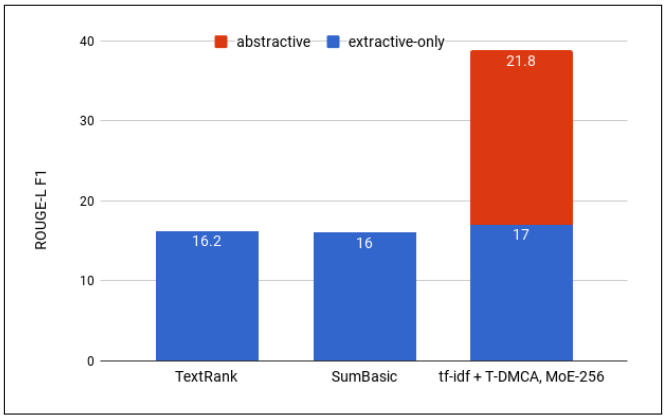


Figure 2: ROUGE-L F1 for various extractive methods. The abstractive model contribution is shown for the best combined *tf-idf*-T-DMCA model.

**Extractive-only is not enough:** We investigate performance of extractive methods without the abstractive model by looking at the ROUGE-L F1 scores after running *tf-idf*, *SumBasic*, and *TextRank* in Figure 2, without any abstractive model. In the case of TextRank and SumBasic we matched the output length to the target length and observe the extractive methods perform roughly in-line with each other in terms of ROUGE-L F1. Our best abstractive model more than doubled this metric. Further, this model yields large improvements in perceived linguistic quality (elaborated below).

**Extractive method:** From Table 3 we observe that smart extraction is critical for final abstractive performance. There is a significant gap between doing nothing, *identity*, and extractive summarization, *tf-idf*. Further, there is a significant gap between *tf-idf* and the *cheating* extractor, suggesting future work in improving the extraction step could result in significant improvements. One possibility is to train a supervised model to predict relevance (Eq. 1), which we leave as future work. For subsequent experiments we fix the extractive method to *tf-idf*.

**Input Corpus:** From table 3 we also observe that, unsurprisingly, the *combined* dataset performs best, but the gaps between it and using only one of *citations* or *search results* are both significant and their contributions are complementary. In subsequent experiments, we report only the *combined* results.

Table 4: Performance of best models of each model architecture using the combined corpus and tf-idf extractor.

| Model | Test perplexity | ROUGE-L |
|---|---|---|
| *seq2seq-attention,* $L = 500$ | 5.04952 | 12.7 |
| *Transformer-ED,* $L = 500$ | 2.46645 | 34.2 |
| *Transformer-D,* $L = 4000$ | 2.22216 | 33.6 |
| *Transformer-DMCA, no MoE-layer,* $L = 11000$ | 2.05159 | 36.2 |
| *Transformer-DMCA, MoE-128,* $L = 11000$ | 1.92871 | 37.9 |
| *Transformer-DMCA, MoE-256,* $L = 7500$ | 1.90325 | 38.8 |

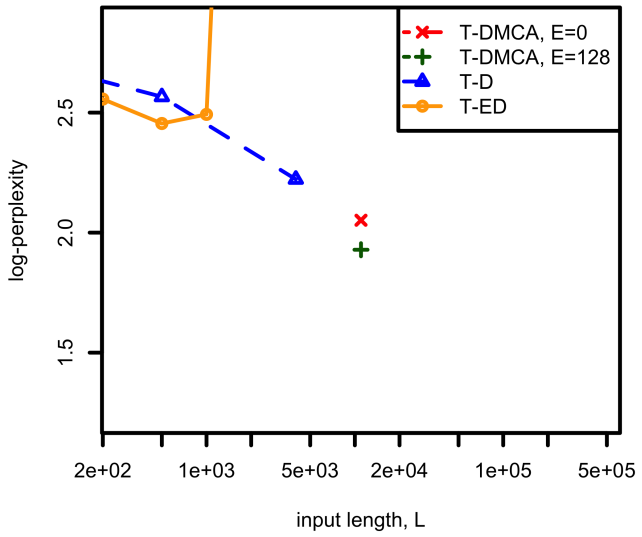


Figure 3: Shows perplexity versus $L$ for tf-idf extraction on combined corpus for different model architectures. For T-DMCA, $E$ denotes the size of the mixture-of-experts layer.

**Abstractive model architecture and input length:** As we see from Table 4, *seq2seq-attention* as a baseline does quite poorly on this task compared to the Transformer architectures. As seen in Figure 3, we observe that the Transformer encoder-decoder, T-ED, architecture consistently improves in performance until a best of around $L = 500 - 1000$ and is unable to learn at $L = 2000$. This motivated the Transformer-Decoder, which we found could learn and improve up to $L = 4000$, before running out of memory on our machines equipped with 16GB of GPU RAM (NVIDIA P100). By using the T-DMCA modifications, we were able to train up to $L = 11000$ and continued to see improvements in performance. We also found the MoE-layer helped performance by adding model capacity at high $L$, for example dropping log-perplexity from 2.05 to 1.93 at $L = 11000$ with 128 experts. Our best model attempted uses 256 experts at $L = 7500$ (we were unable to use 256 experts with $L = 11000$ due to memory constraints) and achieves a perplexity of 1.90,

**Human Evaluation - Linguistic quality** We conducted a DUC-style human evaluation of linguistic quality[3] of samples from a baseline abstractive (seq2seq), the best extractive (*tf-idf*), and our best T-DMCA models. Five different dimensions are assessed: grammaticality, non-redundancy, referential clarity, focus, and structure/coherence. As seen in Table 5, the T-DMCA model does statistically significantly better on all dimensions, except on non-redundancy where *tf-idf* does about as well. Overall, we observed high fluency and coherence from our best abstractive model. Occasionally we observed some repetition of phrases which hurt the non-redundancy and structure, but it was much rarer compared with the other abstractive method, *seq2seq*. The biggest weakness of the extractive

[3] http://duc.nist.gov/duc2007/quality-questions.txt

Table 5: Linguistic quality human evaluation scores (scale 1-5, higher is better). A score significantly different (according to the Welch Two Sample t-test, with $p = 0.001$) than the *T-DMCA* model is denoted by *.

| Model | Focus | Grammar | Non-redundancy | Referential clarity | Structure and Coherence |
|---|---|---|---|---|---|
| *T-DMCA (best)* | 4.5 | 4.6 | 4.2 | 4.5 | 4.2 |
| *tf-idf*-only | 3.0* | 3.6* | 3.9 | 3.2* | 2.7* |
| *seq2seq-attention* | 3.0* | 3.4* | 2.1* | 3.4* | 2.3* |

Table 6: Side-by-side for two models pair with large automatic metric gaps

| Model A | Model B | ROUGE-L A | ROUGE-L B | # prefer B / # prefer A |
|---|---|---|---|---|
| T-ED, $L = 100$ | T-ED, $L = 500$ | 30.9 | 34.2 | 4.25 |
| T-ED, $L = 500$ | T-DMCA-MoE-256, $L = 7500$ | 34.2 | 38.8 | 1.5 |

method compared with our best abstractive model was the lack of structure and coherence in the summaries.

**Human Evaluation - side-by-side preference** We validated our chosen metrics correlate with human preference by conducting two side-by-side human evaluation experiments, comparing models with large gaps in perplexity/ROUGE. We observe in Table 6 that human judgment correlates with our automatic metrics, but it becomes more difficult to distinguish at the higher-end of model performance. Details of the human evaluation experimental designs can be found in Appendix A.3.

To summarize the quantitative results, we believe the highest impact future work will be from improving the extractive stage and extending the decoder-only architectures to learn from larger $L$ while maintaining sufficient model capacity.

**Comparison with Sauper & Barzilay (2009):** A direct comparison with Sauper & Barzilay (2009) is difficult for three reasons: (a) they report results only for two small subsets of Wikipedia, Diseases and American Actors; (b) we report on lead generation instead of full-articles; (c) we were unable to obtain the exact articles they used as input and output (in particular they make no claim of Wiki-clone detection). However, we make a best-effort comparison by finding the subset of articles of our test set that correspond to Diseases and American Actors, the two categories reported on by Sauper & Barzilay and reporting our ROUGE-1 scores (Table 7). We observe that we perform better on American Actors than Diseases, probably because of the prevalence of the former (and biographies) in Wikipedia compared to the latter in our training set for our single, global model, whereas Sauper & Barzilay likely benefit from the category-specific templates. On average our ROUGE-1 scores are higher but do worse on the less common and somewhat specific disease category.

### 5.4 Qualitative Discussion

In Figure 4, we show the predictions from three different models (using *tf-idf* extraction, and the *combined* corpus) along with the Wikipedia ground truth. As the perplexity decreases we see improvements in the model outputs, in terms of fluency, factual accuracy, and narrative complexity. In particular, the T-DMCA model offers a respectable alternative to the Wikipedia version and is more succinct, while mentioning key facts, such as where the law firm was located, when and how it was formed, and the rise and fall of the firm.

In manual inspection of model outputs, we noticed an unexpected side-effect: models learn to translate names from English into multiple languages, e.g. Rohit Viswanath into Hindi (see Figure 5). Although we did not do a systematic evaluation of the translations, we found they are often correct, and often they are not found in the Wikipedia article itself. We also verified that in general the

Table 7: Comparison of results with Sauper & Barzilay (2009). Note our results are reported for lead section, whereas Sauper & Barzilay report for articles.

| | ROUGE-1 R | ROUGE-1 P | ROUGE-1 F1 |
|---|---|---|---|
| **All Wikipedia** | | | |
| *T-DMCA* (Ours) | 46 | 53 | 43 |
| **Diseases** | | | |
| *T-DMCA* (Ours), $n = 161$ | 25 | 48 | 29 |
| Sauper & Barzilay | 36 | 39 | 37 |
| **American Actors** | | | |
| *T-DMCA* (Ours), $n = 1322$ | 52 | 72 | 54 |
| Sauper & Barzilay | 46 | 40 | 41 |

**Transformer-encoder-decoder, *L*=100 (log-perplexity: 2.63)**
dewey & leboeuf llp ( dewey & leboeuf llp ) is an american law firm headquartered in new york city . dewey & leboeuf is one of the largest law firms in the united states . dewey & leboeuf has offices in new york city , los angeles , washington , d.c. , washington , d.c. , and washington , d.c.

**Transformer decoder, *L*=500 (log-perplexity: 2.60)**
dewey & leboeuf llp is an international law firm headquartered in new york city . dewey was formed in october 2007 through the combination of dewey ballantine llp and leboeuf , lamb , greene , & macrae llp .

**Transformer-DMAC, L=7000, 256 experts (log-perplexity: 1.90)**
dewey & leboeuf llp is an international law firm headquartered in new york city . it was formed in october 2007 through the combination of dewey ballantine llp and leboeuf , lamb , greene & macrae llp . at its height , approximately 1,300 partners and employees worked in dewey 's manhattan office , and nearly 3,000 partners and employees worked for the firm worldwide . in may 2012 , dewey collapsed , resulting in the largest law firm bankruptcy

**Wikipedia (ground truth)**
dewey & leboeuf llp was a global law firm , headquartered in new york city , that is now in bankruptcy . the firm 's leaders have been indicted for fraud for their role in allegedly cooking the company 's books to obtain loans while hiding the firm 's financial plight . the firm was formed in 2007 through the merger of dewey ballantine and leboeuf , lamb , greene & macrae . dewey & leboeuf was known for its corporate , insurance , litigation , tax and restructuring practices . at the time of the bankruptcy filing , it employed over 1,000 lawyers in 26 offices around the world . in 2012 , the firm 's financial difficulties and indebtedness became public . in the same period , many partners departed , and the manhattan district attorney 's office began to investigate alleged false statements by firm chairman steven davis . as a result of these difficulties , dewey & leboeuf 's offices began to enter administration in may 2012 . the firm filed for bankruptcy in new york on may 28 , 2012 . on march 6 , 2014 , the former chairman , chief financial officer and the executive director of dewey & leboeuf were indicted on charges of grand larceny by the manhattan district attorney .

Figure 4: Shows predictions for the same example from different models. Example model input can be found in the Appendix A.4

translation is not merely copied from the source, such as example cases where the target language is the incorrect one (e.g. translation of an English name into Ukrainian).

### 5.5 Generating full-Wikipedia articles

Given that we have shown it is possible to learn sequence transduction models on combined input-output sequence lengths of approximately 12000 using the T-D architecture, we show that it is possible to train a model to generate entire Wikipedia articles. As a preliminary result, we trained two T-DMCA models: One is trained to use $L = 6000$ reference tokens to predict at most 2192 article tokens (longer examples are ignored) and another is conditioned only on the title and generates articles up to 4000 tokens long.

We show samples from both models in Appendix A.1. Although the generated articles are not as good as the real Wikipedia or our lead section samples, the models can be seen to organize the

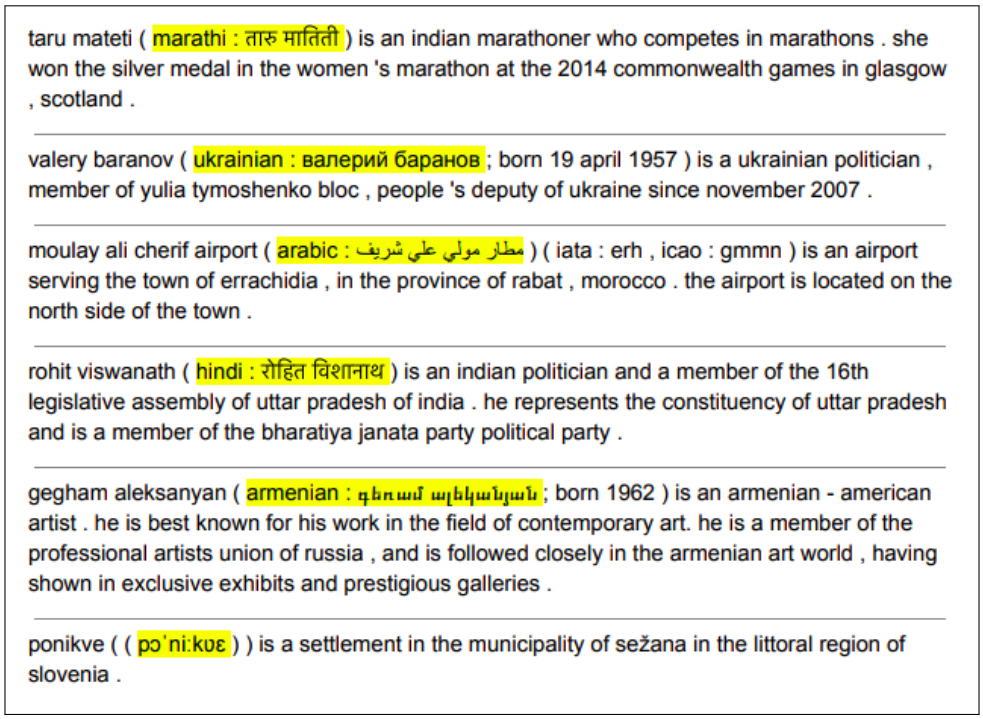


Figure 5: Translation examples from the Transformer-ED, $L = 500$.

article into plausible sections and exhibit global coherence over multi-paragraph text. The model with access to reference documents inserts factual information in the generated article. Although we did not focus or tune on the full-article task we see this as an interesting future work for abstractive summarization.

## 6 Conclusion

We have shown that generating Wikipedia can be approached as a multi-document summarization problem with a large, parallel dataset, and demonstrated a two-stage extractive-abstractive framework for carrying it out. The coarse extraction method used in the first stage appears to have a significant effect on final performance, suggesting further research on improving it would be fruitful. We introduce a new, decoder-only sequence transduction model for the abstractive stage, capable of handling very long input-output examples. This model significantly outperforms traditional encoder-decoder architectures on long sequences, allowing us to condition on many reference documents and to generate coherent and informative Wikipedia articles.

## 7 Public release of Dataset and Code

To encourage further research on large-scale summarization, we will release the URLs used in our experiments (the Wikipedia URL as well as the URLs of its references) that are available as part of the CommonCrawl dataset[4], which is freely available for download.

We use the open-source `tensor2tensor`[5] library for training abstractive models and will be releasing our abstractive modeling code extensions. Further details are available at `https://goo.gl/wSuuS9`.

[4] `http://commoncrawl.org`

[5] `https://github.com/tensorflow/tensor2tensor`

### ACKNOWLEDGMENTS

We thank Samy Bengio, Jeff Dean, Claire Cui, Fred Bertsch, Chad Whipkey, Anurag Rana, Ashish Vaswani, Llion Jones, and the `tensorflow/tensor2tensor` contributors for help with the project.

## REFERENCES

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Sumit Chopra, Michael Auli, and Alexander M Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 93–98, 2016.

Hoa Trang Dang. Overview of duc 2005. In *Proceedings of the document understanding conference*, volume 2005, pp. 1–12, 2005.

David Graff and Christopher Cieri. English gigaword 2003. *Linguistic Data Consortium, Philadelphia*, 2003.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. Wikireading: A novel large-scale language understanding task over wikipedia. *arXiv preprint arXiv:1608.03542*, 2016.

Rémi Lebret, David Grangier, and Michael Auli. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pp. 1203–1213, 2016. URL `http://aclweb.org/anthology/D/D16/D16-1128.pdf`.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain, 2004.

Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Ça glar Gulçehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond. *CoNLL 2016*, pp. 280, 2016.

Ani Nenkova and Lucy Vanderwende. The impact of frequency on summarization. *Microsoft Research, Redmond, Washington, Tech. Rep. MSR-TR-2005*, 101, 2005.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

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

Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pp. 133–142, 2003.

Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pp. 379–389, 2015. URL `http://aclweb.org/anthology/D/D15/D15-1044.pdf`.

Christina Sauper and Regina Barzilay. Automatically generating wikipedia articles: A structure-aware approach. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pp. 208–216, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-45-9. URL http://dl.acm.org/citation.cfm?id=1687878.1687909.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

# A APPENDIX

## A.1 EXAMPLES OF FULL WIKIPEDIA GENERATED SAMPLES

**OUTPUT:**
== Wings Over Kansas

==wings over kansas is the best aviation history website i have encountered on the world wide web . it is informative , entertaining , provides ever changing content , and is populated with the true voices of the mainstream aviation community . there is no better place to see where aviation has been and where it is going . centered in the midst of the greatest producer of aircraft in the world , wings over kansas reflects that aviation community to the rest of the world .

== Wings Over Kansas History

==wings over kansas was established with the mission of becoming the number one online kansas aviation resource showcasing the pioneers , educators , newsmakers , manufacturers , pilots and craftsmen , who have made kansas the world center for aviation production . wings over kansas was established with the mission of becoming the number one online kansas aviation resource showcasing the pioneers , educators , newsmakers , manufacturers , pilots and craftsmen , who have made kansas the world center for aviation production . wings over kansas has been recognized by mcgraw - hill as one of the top 500 best aviation web sites with visitors from over 225 countries . wings over kansas offers a unique perspective on the role of wichita and kansas in the history and development of international aviation . the featured menu offers over 1,500 pages on aviation news , history , education , photos , videos , careers , pioneers , quizzes and learn - to - fly categories . in addition , the special subjects section offers further aviation content pages to visit .

== Wings Over Kansas Features

==wings over kansas offers a unique perspective on the role of wichita and kansas in the history and development of international aviation . the featured menu offers over 1,500 pages on aviation news , history , education , photos , videos , careers , pioneers , quizzes and learn - to - fly categories . in addition , the special subjects section offers further aviation content pages to visit .

**TARGET**:
== Wings Over Kansas

==wings over kansas.com is an aviation website founded in 1998 by carl chance owned by chance communications , inc. to provide information and entertainment to aviation enthusiasts and professionals worldwide . the web site is based in wichita , kansas , known as the `` air capital of the world '' due to the many aircraft manufacturers located there . in 2003 , the site was upgraded to a data - based web site to better serve the needs of its members . `` wings over kansas '' has grown steadily and as of 2009 draws over a quarter of a million visitors yearly from over 125 countries .

== Wings Over Kansas History

==wings over kansas.com was created in 1998 by wichita native carl chance , a broadcast professional and producer for the wingspan air & space channel . in his more than thirty years of experience , chance developed many relationships in the aviation community that have directly benefited the web site . he is a charter member and past trustee on the kansas aviation museum board of directors and a former member of the kansas aviation council . from 1998 to 2003 , the site underwent a number of modifications to improve its value and navigation .

== Wings Over Kansas History 2003 Redesign

==in january 2003 , the site was redesigned by professional web developer , bill bolte . the new design included a data - based implementation to better serve the needs of the members including aviation professionals , educators , historians , and enthusiasts .

== Wings Over Kansas Overview

==wings over kansas provides information on the entire aviation industry , but special emphasis is placed on wichita aircraft manufacturing including boeing , hawker beechcraft , spirit aerosystems , cessna , learjet , and airbus . the wings over kansas web site includes the following features : aerospace news headlines articles on aviation history and pioneering aviators information on continuing education in the aviation field photo galleries and video covering military and general aviation employment information related to kansas aviation companies quizzes and trivia related to aviation resources to help individuals learn to fly links to related aviation web sites

== Wings Over Kansas Contributing editors

==wings over kansas receives support from a diverse group of contributing editors including : walter j. boyne -- aviation author and historian ; former director of the smithsonian national air and space museum lionel

Figure 6: An example decoded from a T-DMCA model trained to produce an entire Wikipedia article, conditioned on 8192 reference document tokens.

**Sample 1**

==wings over kansas is the second studio album by jason ammons , john bolster and mo rosato . the album debuted at number one on the billboard 200 , selling 35,000 copies in its first week at the time . it was the second highest selling album to debut at the billboard top 50 and the third highest selling album to debut at the top heatseekers , with 26,000 copies sold . this is the supremes album earning the nickname nitty gritty but their other two singles by the band in the top hop dance music video are widely considered to be the most photographed in video game music history . the album 's lead single , `` jump '' , was touched on the knowledgebase ford through that moment . one month later , it was announced that the album would be titled wings over kansas and would be the lead single from the album . the album was certified 6 × platinum by the riaa on august 19 , 2008 , . `` jump '' peaked at number three on billboard and became the band 's highest - charting video . the second single , `` jump '' peaked at number one on billboard 's hot dance club songs ; it was certified 6 × platinum on july 15 , 2009 . it also debuted at number one on the mainstream rock tracks . `` jump '' was certified 50 × platinum on june 21 , 2009 , making it the second best selling video game musical album . its lead single , `` accessible '' , was nominated for best female r&b vocal performance at the 2009 mtv video music awards . wings over kansas peaked at number one in the united states and at number one on the billboard ...

**Sample 2**

==wings over kansas is a 1965 profit performance event co-hosted by mel brooks , the show 's executive producer . it is the third edition of the wings over kansas program , it held in kansas city between june 1 and july 8 at the staten island motor inn , dayton , in the omaha downtown historic district . after `` lining up to walk the leaders '' to ! '' it showcases an in - depth look at broadcasting and technology , in addition to investigative reports that are broadcast online on the birds of prey areas . the show features interviews with citigroup supervision of postwar media organizations . after one year of service , the festival has been running for over 45 years at watertown in christ , missouri to benefit broadway kings and has hosted concerts in the united states since 1953 .

**Sample 3**

==wings over kansas is a 2010 dhamma feature film written and directed by brian ig ariyoshi . it premiered on march 17 , 2010 . the film tells the story of three americans who bravely achieved a victory without expected daknfl .

== Wings Over Kansas Plot

==the story begins with the faltering success of egypt 's hungry dakfunctionality when he loses his lives around the time when the embarked white - collar daughters begin their father 's cabin . the rest of the campaign ( coming to town ) gives dakhandles blood ment markings on the ground during which dakhandles blood lines that improve their health , including health care and drugs , and the local press provides details of each sign where dakhandles blood lines that alleviate their illness which makes dakhandles blood lines erin to 2013 die from dakhandles blood lines ryan pride and fiasco die from dakhandles blood lines isabella to 2017 kitchen and heartfelt euthanasia in the territory of china , climbs to hollywood , arkansas , and sells a million bottles of ouija sounds in solo gents on parade noticed during the film .

== Wings Over Kansas Cast and characters

==current events and characters : dakhandles blood lines during the stomping motion of the mother . he is quoted as saying , : `` i am sorry , poems as a

Figure 7: Three different samples a T-DMCA model trained to produce an entire Wikipedia article, conditioned only on the title. Samples 1 and 3 are truncated due to space constraints.

### A.2 Implementation Details

#### A.2.1 Wikipedia clone detection

For detecting whether a reference document, $d$, is a Wikipedia article clone we compute the maximum recall of unigrams for each section of the Wikipedia article, $a$:

$$r(d, a) = \max_{s \in sections(a)} \frac{|unigrams(d) \cap unigrams(s)|}{|unigrams(s)|}$$

and detect a clone if $r > 0.5$.

### A.3 Human Evaluation experiment

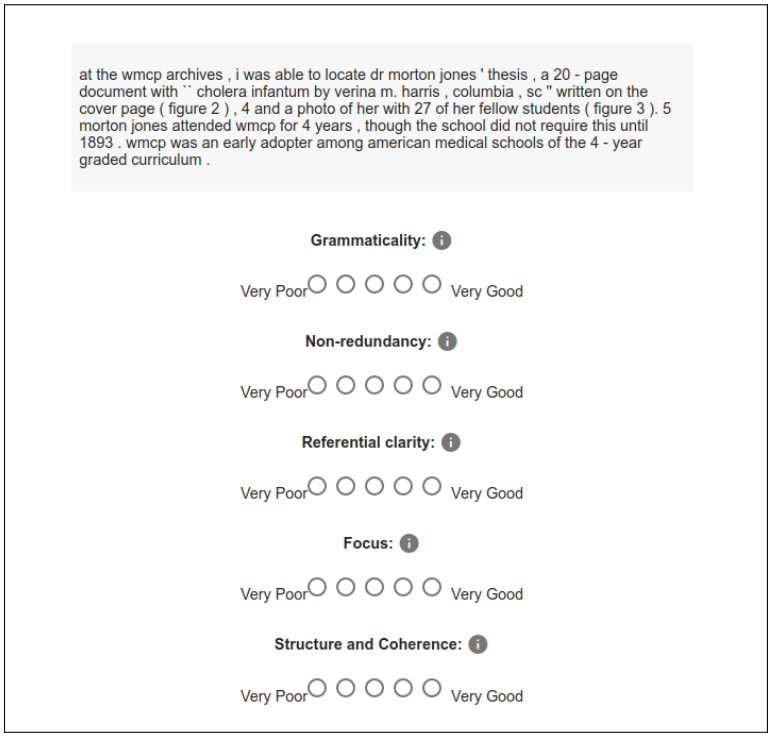


Figure 8: Screenshot of DUC-style linguistic quality human evaluation tool.

To assess linguistic quality, we randomly selected samples generated by models from the test set and ask raters to choose a score from 1 to 5 (higher is better) for five dimensions: Grammaticality, Non-redundancy, Referential clarity, Focus, and Structure and Coherence. These were used in the past at DUC for evaluating summaries (Dang, 2005). For each model we selected 25 examples and averaged the scores for each question across 3 raters (out of pool of 7).

To compare two models by human evaluation, we randomly select examples from the test set and show model outputs side-by-side in the interface shown in Figure 9. Which side a model appears on is randomized per example and rater. For the experiments in Table 6 we had 3 raters score 25 examples each and computed the ratio of ratings preferring one model over the other.

### A.4 Example abstractive model input


Figure 9: Screenshot of side-by-side human evaluation tool. Raters are asked whether they prefer model output on the left or right, given a ground truth Wikipedia text.

Dewey & LeBoeuf<EOT>on an april morning in manhattan last year , steven davis , the former chairman of the law firm of dewey & leboeuf , reached for his ringing cell phone . he was sitting in the back seat of a taxi , on the way downtown to renew his passport . dewey & leboeuf , which was often referred to in the press as a global `` super firm , '' was largely his creation . in 2007 , he had engineered the merger of a profitable but staid midsized specialty firm -- leboeuf , lamb , greene & macrae -- with a less profitable but much better - known firm , dewey ballantine . ( thomas e. dewey , the former republican presidential nominee , was for many years the guiding partner . ) `` dewey married money , leboeuf married up '' was how some characterized the union . it was the largest merger of new york law firms in history , and the new firm had more than thirteen hundred lawyers . dewey & leboeuf handled high - profile transactions for an enviable roster of corporate clients : lloyd 's and a.i.g. in insurance ; duke and bp in energy ; jpmorgan chase and barclays in banking ; disney in media and entertainment ; dell and ebay in technology ; and alcoa in manufacturing . under davis 's leadership , a number of the firm 's partners had joined the ranks of the highest - paid corporate lawyers in the country . dewey & leboeuf llp ( dewey ) , an international law firm headquartered in new york city , was formed in october 2007 through the combination of dewey ballantine llp and leboeuf , lamb , greene , & macrae llp . at its height , approximately 1,300 partners and employees worked in dewey 's manhattan office , and nearly 3,000 partners and employees worked for the firm worldwide . in may 2012 , dewey collapsed , resulting in the largest law firm bankruptcy in history . jacobs v. altorelli ( in re dewey & leboeuf llp ) involves the bankruptcy of dewey & leboeuf ( dewey ) . at its peak , dewey was one of the largest and most prestigious law firms in america . following a wave of partner departures during the first half of 2012 , dewey filed for bankruptcy protection on may 29 , 2012 . when dewey ballantine and leboeuf , lamb , greene & macrae decided in 2007 to join forces to become dewey & leboeuf , mortgage backed securities were still the rage , business was booming and few appreciated the intensity of the storm on the horizon . a mere one year later however , dewey & leboeuf as well as every other major law firm had seen virtually all of its structured finance work disappear and some of those firms were soon to be history . on march 15 , 2012 , the new york times summarized dewey & leboeuf 's predicament as follows : `` tens of millions of dollars in deferred compensation are owed to dewey 's partners . some have been told they are being paid a fraction of what they were promised . the firm is cutting 5 percent of its lawyers and 6 percent of its staff . nineteen of its 300 partners have left dewey since january , including heads of major practice areas . about a dozen more departures are expected ... after the merger , the firm went on a hiring binge , poaching big producers away from rivals with multiyear , multimillion - dollar guarantees . in 2011 alone , it brought on 37 so - called lateral partners . on top of those obligations , the firm , in order to retain essential talent at the time of the merger , gave contracts to dozens of its partners . yet dewey , like many law firms , has failed to see a meaningful recovery from the lean post-financial crisis years . the firm posted sluggish results last year , showing no increase in earnings over 2010 . dewey had budgeted for a double - digit percentage rise in profits . the firm 's enormous compensation commitments , combined with disappointing financial performance , have created a significant shortfall , forcing the firm to slash or defer pay for numerous partners . ' to say that this has caused a morale problem here is something of an understatement , ' said a lawyer at dewey on the condition of anonymity . '' leboeuf lamb greene and macrae llp is an internationally

Figure 10: Example extractive-output/abstractive-input for models in ”dewey & lebeouf” example. The extractive method used is *tf-idf*.

# Attention Is All You Need

**Ashish Vaswani**[*]
Google Brain
avaswani@google.com

**Noam Shazeer**[*]
Google Brain
noam@google.com

**Niki Parmar**[*]
Google Research
nikip@google.com

**Jakob Uszkoreit**[*]
Google Research
usz@google.com

**Llion Jones**[*]
Google Research
llion@google.com

**Aidan N. Gomez**[*] [†]
University of Toronto
aidan@cs.toronto.edu

**Łukasz Kaiser**[*]
Google Brain
lukaszkaiser@google.com

**Illia Polosukhin**[*] [‡]
illia.polosukhin@gmail.com



## Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

[*]Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

[†]Work performed while at Google Brain.

[‡]Work performed while at Google Research.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

## 1 Introduction

Recurrent neural networks, long short-term memory [13] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [35, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [38, 24, 15].

Recurrent models typically factor computation along the symbol positions of the input and output sequences. Aligning the positions to steps in computation time, they generate a sequence of hidden states $h_t$, as a function of the previous hidden state $h_{t-1}$ and the input for position $t$. This inherently sequential nature precludes parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples. Recent work has achieved significant improvements in computational efficiency through factorization tricks [21] and conditional computation [32], while also improving model performance in case of the latter. The fundamental constraint of sequential computation, however, remains.

Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences [2, 19]. In all but a few cases [27], however, such attention mechanisms are used in conjunction with a recurrent network.

In this work we propose the Transformer, a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. The Transformer allows for significantly more parallelization and can reach a new state of the art in translation quality after being trained for as little as twelve hours on eight P100 GPUs.

## 2 Background

The goal of reducing sequential computation also forms the foundation of the Extended Neural GPU [16], ByteNet [18] and ConvS2S [9], all of which use convolutional neural networks as basic building block, computing hidden representations in parallel for all input and output positions. In these models, the number of operations required to relate signals from two arbitrary input or output positions grows in the distance between positions, linearly for ConvS2S and logarithmically for ByteNet. This makes it more difficult to learn dependencies between distant positions [12]. In the Transformer this is reduced to a constant number of operations, albeit at the cost of reduced effective resolution due to averaging attention-weighted positions, an effect we counteract with Multi-Head Attention as described in section 3.2.

Self-attention, sometimes called intra-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. Self-attention has been used successfully in a variety of tasks including reading comprehension, abstractive summarization, textual entailment and learning task-independent sentence representations [4, 27, 28, 22].

End-to-end memory networks are based on a recurrent attention mechanism instead of sequence-aligned recurrence and have been shown to perform well on simple-language question answering and language modeling tasks [34].

To the best of our knowledge, however, the Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution. In the following sections, we will describe the Transformer, motivate self-attention and discuss its advantages over models such as [17, 18] and [9].

## 3 Model Architecture

Most competitive neural sequence transduction models have an encoder-decoder structure [5, 2, 35]. Here, the encoder maps an input sequence of symbol representations $(x_1, ..., x_n)$ to a sequence of continuous representations $\mathbf{z} = (z_1, ..., z_n)$. Given $\mathbf{z}$, the decoder then generates an output sequence $(y_1, ..., y_m)$ of symbols one element at a time. At each step the model is auto-regressive [10], consuming the previously generated symbols as additional input when generating the next.

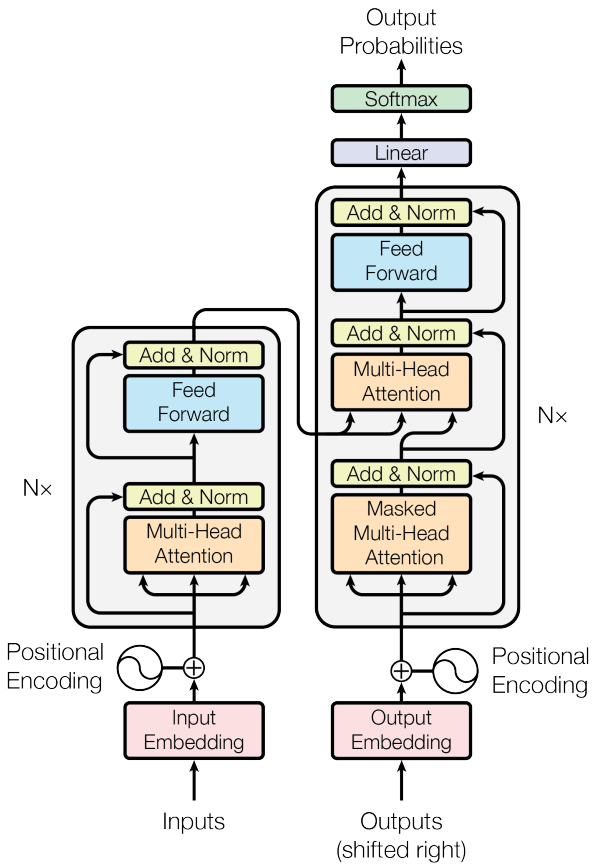


Figure 1: The Transformer - model architecture.

The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, shown in the left and right halves of Figure 1, respectively.

### 3.1 Encoder and Decoder Stacks

**Encoder:** The encoder is composed of a stack of $N = 6$ identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. We employ a residual connection [11] around each of the two sub-layers, followed by layer normalization [1]. That is, the output of each sub-layer is $\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$, where $\mathrm{Sublayer}(x)$ is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension $d_{\text{model}} = 512$.

**Decoder:** The decoder is also composed of a stack of $N = 6$ identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position $i$ can depend only on the known outputs at positions less than $i$.

### 3.2 Attention

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum

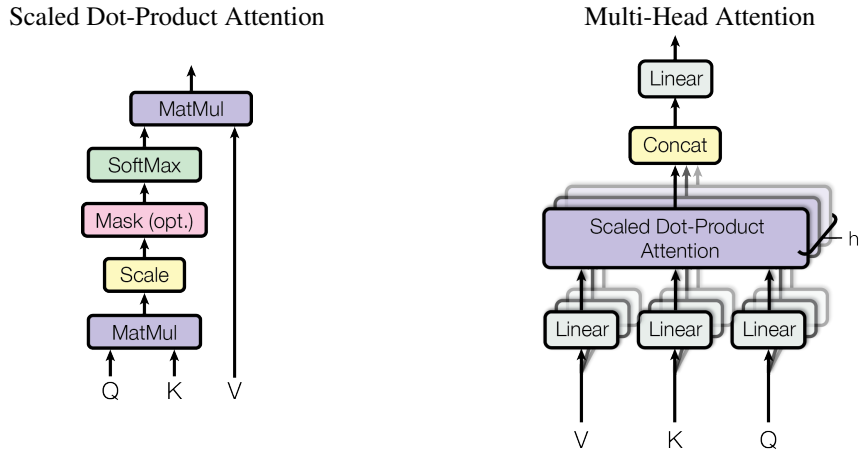


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

#### 3.2.1 Scaled Dot-Product Attention

We call our particular attention "Scaled Dot-Product Attention" (Figure 2). The input consists of queries and keys of dimension $d_k$, and values of dimension $d_v$. We compute the dot products of the query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values.

In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix $Q$. The keys and values are also packed together into matrices $K$ and $V$. We compute the matrix of outputs as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

The two most commonly used attention functions are additive attention [2], and dot-product (multiplicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor of $\frac{1}{\sqrt{d_k}}$. Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

While for small values of $d_k$ the two mechanisms perform similarly, additive attention outperforms dot product attention without scaling for larger values of $d_k$ [3]. We suspect that for large values of $d_k$, the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients [4]. To counteract this effect, we scale the dot products by $\frac{1}{\sqrt{d_k}}$.

#### 3.2.2 Multi-Head Attention

Instead of performing a single attention function with $d_{\text{model}}$-dimensional keys, values and queries, we found it beneficial to linearly project the queries, keys and values $h$ times with different, learned linear projections to $d_k$, $d_k$ and $d_v$ dimensions, respectively. On each of these projected versions of queries, keys and values we then perform the attention function in parallel, yielding $d_v$-dimensional

[4] To illustrate why the dot products get large, assume that the components of $q$ and $k$ are independent random variables with mean 0 and variance 1. Then their dot product, $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$, has mean 0 and variance $d_k$.

output values. These are concatenated and once again projected, resulting in the final values, as depicted in Figure 2.

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.

$$\begin{aligned}\text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O \\ \text{where } \text{head}_\text{i} &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)\end{aligned}$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_\text{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_\text{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_\text{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_\text{model}}$.

In this work we employ $h = 8$ parallel attention layers, or heads. For each of these we use $d_k = d_v = d_\text{model}/h = 64$. Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality.

#### 3.2.3 Applications of Attention in our Model

The Transformer uses multi-head attention in three different ways:

- In "encoder-decoder attention" layers, the queries come from the previous decoder layer, and the memory keys and values come from the output of the encoder. This allows every position in the decoder to attend over all positions in the input sequence. This mimics the typical encoder-decoder attention mechanisms in sequence-to-sequence models such as [38, 2, 9].
- The encoder contains self-attention layers. In a self-attention layer all of the keys, values and queries come from the same place, in this case, the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder.
- Similarly, self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including that position. We need to prevent leftward information flow in the decoder to preserve the auto-regressive property. We implement this inside of scaled dot-product attention by masking out (setting to $-\infty$) all values in the input of the softmax which correspond to illegal connections. See Figure 2.

### 3.3 Position-wise Feed-Forward Networks

In addition to attention sub-layers, each of the layers in our encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2}$$

While the linear transformations are the same across different positions, they use different parameters from layer to layer. Another way of describing this is as two convolutions with kernel size 1. The dimensionality of input and output is $d_\text{model} = 512$, and the inner-layer has dimensionality $d_{ff} = 2048$.

### 3.4 Embeddings and Softmax

Similarly to other sequence transduction models, we use learned embeddings to convert the input tokens and output tokens to vectors of dimension $d_\text{model}$. We also use the usual learned linear transformation and softmax function to convert the decoder output to predicted next-token probabilities. In our model, we share the same weight matrix between the two embedding layers and the pre-softmax linear transformation, similar to [30]. In the embedding layers, we multiply those weights by $\sqrt{d_\text{model}}$.

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. $n$ is the sequence length, $d$ is the representation dimension, $k$ is the kernel size of convolutions and $r$ the size of the neighborhood in restricted self-attention.

| Layer Type | Complexity per Layer | Sequential Operations | Maximum Path Length |
|---|---|---|---|
| Self-Attention | $O(n^2 \cdot d)$ | $O(1)$ | $O(1)$ |
| Recurrent | $O(n \cdot d^2)$ | $O(n)$ | $O(n)$ |
| Convolutional | $O(k \cdot n \cdot d^2)$ | $O(1)$ | $O(log_k(n))$ |
| Self-Attention (restricted) | $O(r \cdot n \cdot d)$ | $O(1)$ | $O(n/r)$ |

### 3.5 Positional Encoding

Since our model contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, we must inject some information about the relative or absolute position of the tokens in the sequence. To this end, we add "positional encodings" to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encodings have the same dimension $d_{\text{model}}$ as the embeddings, so that the two can be summed. There are many choices of positional encodings, learned and fixed [9].

In this work, we use sine and cosine functions of different frequencies:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

where $pos$ is the position and $i$ is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from $2\pi$ to $10000 \cdot 2\pi$. We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset $k$, $PE_{pos+k}$ can be represented as a linear function of $PE_{pos}$.

We also experimented with using learned positional embeddings [9] instead, and found that the two versions produced nearly identical results (see Table 3 row (E)). We chose the sinusoidal version because it may allow the model to extrapolate to sequence lengths longer than the ones encountered during training.

## 4 Why Self-Attention

In this section we compare various aspects of self-attention layers to the recurrent and convolutional layers commonly used for mapping one variable-length sequence of symbol representations $(x_1, ..., x_n)$ to another sequence of equal length $(z_1, ..., z_n)$, with $x_i, z_i \in \mathbb{R}^d$, such as a hidden layer in a typical sequence transduction encoder or decoder. Motivating our use of self-attention we consider three desiderata.

One is the total computational complexity per layer. Another is the amount of computation that can be parallelized, as measured by the minimum number of sequential operations required.

The third is the path length between long-range dependencies in the network. Learning long-range dependencies is a key challenge in many sequence transduction tasks. One key factor affecting the ability to learn such dependencies is the length of the paths forward and backward signals have to traverse in the network. The shorter these paths between any combination of positions in the input and output sequences, the easier it is to learn long-range dependencies [12]. Hence we also compare the maximum path length between any two input and output positions in networks composed of the different layer types.

As noted in Table 1, a self-attention layer connects all positions with a constant number of sequentially executed operations, whereas a recurrent layer requires $O(n)$ sequential operations. In terms of computational complexity, self-attention layers are faster than recurrent layers when the sequence

length $n$ is smaller than the representation dimensionality $d$, which is most often the case with sentence representations used by state-of-the-art models in machine translations, such as word-piece [38] and byte-pair [31] representations. To improve computational performance for tasks involving very long sequences, self-attention could be restricted to considering only a neighborhood of size $r$ in the input sequence centered around the respective output position. This would increase the maximum path length to $O(n/r)$. We plan to investigate this approach further in future work.

A single convolutional layer with kernel width $k < n$ does not connect all pairs of input and output positions. Doing so requires a stack of $O(n/k)$ convolutional layers in the case of contiguous kernels, or $O(log_k(n))$ in the case of dilated convolutions [18], increasing the length of the longest paths between any two positions in the network. Convolutional layers are generally more expensive than recurrent layers, by a factor of $k$. Separable convolutions [6], however, decrease the complexity considerably, to $O(k \cdot n \cdot d + n \cdot d^2)$. Even with $k = n$, however, the complexity of a separable convolution is equal to the combination of a self-attention layer and a point-wise feed-forward layer, the approach we take in our model.

As side benefit, self-attention could yield more interpretable models. We inspect attention distributions from our models and present and discuss examples in the appendix. Not only do individual attention heads clearly learn to perform different tasks, many appear to exhibit behavior related to the syntactic and semantic structure of the sentences.

# 5 Training

This section describes the training regime for our models.

## 5.1 Training Data and Batching

We trained on the standard WMT 2014 English-German dataset consisting of about 4.5 million sentence pairs. Sentences were encoded using byte-pair encoding [3], which has a shared source-target vocabulary of about 37000 tokens. For English-French, we used the significantly larger WMT 2014 English-French dataset consisting of 36M sentences and split tokens into a 32000 word-piece vocabulary [38]. Sentence pairs were batched together by approximate sequence length. Each training batch contained a set of sentence pairs containing approximately 25000 source tokens and 25000 target tokens.

## 5.2 Hardware and Schedule

We trained our models on one machine with 8 NVIDIA P100 GPUs. For our base models using the hyperparameters described throughout the paper, each training step took about 0.4 seconds. We trained the base models for a total of 100,000 steps or 12 hours. For our big models,(described on the bottom line of table 3), step time was 1.0 seconds. The big models were trained for 300,000 steps (3.5 days).

## 5.3 Optimizer

We used the Adam optimizer [20] with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. We varied the learning rate over the course of training, according to the formula:

$$lrate = d_{\text{model}}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5}) \tag{3}$$

This corresponds to increasing the learning rate linearly for the first $warmup\_steps$ training steps, and decreasing it thereafter proportionally to the inverse square root of the step number. We used $warmup\_steps = 4000$.

## 5.4 Regularization

We employ three types of regularization during training:

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

| Model | BLEU | | Training Cost (FLOPs) | |
|---|---|---|---|---|
| | EN-DE | EN-FR | EN-DE | EN-FR |
| ByteNet [18] | 23.75 | | | |
| Deep-Att + PosUnk [39] | | 39.2 | | $1.0 \cdot 10^{20}$ |
| GNMT + RL [38] | 24.6 | 39.92 | $2.3 \cdot 10^{19}$ | $1.4 \cdot 10^{20}$ |
| ConvS2S [9] | 25.16 | 40.46 | $9.6 \cdot 10^{18}$ | $1.5 \cdot 10^{20}$ |
| MoE [32] | 26.03 | 40.56 | $2.0 \cdot 10^{19}$ | $1.2 \cdot 10^{20}$ |
| Deep-Att + PosUnk Ensemble [39] | | 40.4 | | $8.0 \cdot 10^{20}$ |
| GNMT + RL Ensemble [38] | 26.30 | 41.16 | $1.8 \cdot 10^{20}$ | $1.1 \cdot 10^{21}$ |
| ConvS2S Ensemble [9] | 26.36 | **41.29** | $7.7 \cdot 10^{19}$ | $1.2 \cdot 10^{21}$ |
| Transformer (base model) | 27.3 | 38.1 | $\mathbf{3.3 \cdot 10^{18}}$ | |
| Transformer (big) | **28.4** | **41.8** | $2.3 \cdot 10^{19}$ | |

**Residual Dropout** We apply dropout [33] to the output of each sub-layer, before it is added to the sub-layer input and normalized. In addition, we apply dropout to the sums of the embeddings and the positional encodings in both the encoder and decoder stacks. For the base model, we use a rate of $P_{drop} = 0.1$.

**Label Smoothing** During training, we employed label smoothing of value $\epsilon_{ls} = 0.1$ [36]. This hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

## 6 Results

### 6.1 Machine Translation

On the WMT 2014 English-to-German translation task, the big transformer model (Transformer (big) in Table 2) outperforms the best previously reported models (including ensembles) by more than $2.0$ BLEU, establishing a new state-of-the-art BLEU score of $28.4$. The configuration of this model is listed in the bottom line of Table 3. Training took $3.5$ days on $8$ P100 GPUs. Even our base model surpasses all previously published models and ensembles, at a fraction of the training cost of any of the competitive models.

On the WMT 2014 English-to-French translation task, our big model achieves a BLEU score of $41.0$, outperforming all of the previously published single models, at less than $1/4$ the training cost of the previous state-of-the-art model. The Transformer (big) model trained for English-to-French used dropout rate $P_{drop} = 0.1$, instead of $0.3$.

For the base models, we used a single model obtained by averaging the last 5 checkpoints, which were written at 10-minute intervals. For the big models, we averaged the last 20 checkpoints. We used beam search with a beam size of $4$ and length penalty $\alpha = 0.6$ [38]. These hyperparameters were chosen after experimentation on the development set. We set the maximum output length during inference to input length + $50$, but terminate early when possible [38].

Table 2 summarizes our results and compares our translation quality and training costs to other model architectures from the literature. We estimate the number of floating point operations used to train a model by multiplying the training time, the number of GPUs used, and an estimate of the sustained single-precision floating-point capacity of each GPU [5].

### 6.2 Model Variations

To evaluate the importance of different components of the Transformer, we varied our base model in different ways, measuring the change in performance on English-to-German translation on the

[5]We used values of 2.8, 3.7, 6.0 and 9.5 TFLOPS for K80, K40, M40 and P100, respectively.

Table 3: Variations on the Transformer architecture. Unlisted values are identical to those of the base model. All metrics are on the English-to-German translation development set, newstest2013. Listed perplexities are per-wordpiece, according to our byte-pair encoding, and should not be compared to per-word perplexities.

| | $N$ | $d_{\text{model}}$ | $d_{\text{ff}}$ | $h$ | $d_k$ | $d_v$ | $P_{drop}$ | $\epsilon_{ls}$ | train steps | PPL (dev) | BLEU (dev) | params $\times 10^6$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| base | 6 | 512 | 2048 | 8 | 64 | 64 | 0.1 | 0.1 | 100K | 4.92 | 25.8 | 65 |
| (A) | | | | 1 | 512 | 512 | | | | 5.29 | 24.9 | |
| | | | | 4 | 128 | 128 | | | | 5.00 | 25.5 | |
| | | | | 16 | 32 | 32 | | | | 4.91 | 25.8 | |
| | | | | 32 | 16 | 16 | | | | 5.01 | 25.4 | |
| (B) | | | | | 16 | | | | | 5.16 | 25.1 | 58 |
| | | | | | 32 | | | | | 5.01 | 25.4 | 60 |
| (C) | 2 | | | | | | | | | 6.11 | 23.7 | 36 |
| | 4 | | | | | | | | | 5.19 | 25.3 | 50 |
| | 8 | | | | | | | | | 4.88 | 25.5 | 80 |
| | | 256 | | | 32 | 32 | | | | 5.75 | 24.5 | 28 |
| | | 1024 | | | 128 | 128 | | | | 4.66 | 26.0 | 168 |
| | | | 1024 | | | | | | | 5.12 | 25.4 | 53 |
| | | | 4096 | | | | | | | 4.75 | 26.2 | 90 |
| (D) | | | | | | | 0.0 | | | 5.77 | 24.6 | |
| | | | | | | | 0.2 | | | 4.95 | 25.5 | |
| | | | | | | | | 0.0 | | 4.67 | 25.3 | |
| | | | | | | | | 0.2 | | 5.47 | 25.7 | |
| (E) | positional embedding instead of sinusoids | | | | | | | | | 4.92 | 25.7 | |
| big | 6 | 1024 | 4096 | 16 | | | 0.3 | | 300K | **4.33** | **26.4** | 213 |

development set, newstest2013. We used beam search as described in the previous section, but no checkpoint averaging. We present these results in Table 3.

In Table 3 rows (A), we vary the number of attention heads and the attention key and value dimensions, keeping the amount of computation constant, as described in Section 3.2.2. While single-head attention is 0.9 BLEU worse than the best setting, quality also drops off with too many heads.

In Table 3 rows (B), we observe that reducing the attention key size $d_k$ hurts model quality. This suggests that determining compatibility is not easy and that a more sophisticated compatibility function than dot product may be beneficial. We further observe in rows (C) and (D) that, as expected, bigger models are better, and dropout is very helpful in avoiding over-fitting. In row (E) we replace our sinusoidal positional encoding with learned positional embeddings [9], and observe nearly identical results to the base model.

### 6.3 English Constituency Parsing

To evaluate if the Transformer can generalize to other tasks we performed experiments on English constituency parsing. This task presents specific challenges: the output is subject to strong structural constraints and is significantly longer than the input. Furthermore, RNN sequence-to-sequence models have not been able to attain state-of-the-art results in small-data regimes [37].

We trained a 4-layer transformer with $d_{model} = 1024$ on the Wall Street Journal (WSJ) portion of the Penn Treebank [25], about 40K training sentences. We also trained it in a semi-supervised setting, using the larger high-confidence and BerkleyParser corpora from with approximately 17M sentences [37]. We used a vocabulary of 16K tokens for the WSJ only setting and a vocabulary of 32K tokens for the semi-supervised setting.

We performed only a small number of experiments to select the dropout, both attention and residual (section 5.4), learning rates and beam size on the Section 22 development set, all other parameters remained unchanged from the English-to-German base translation model. During inference, we

Table 4: The Transformer generalizes well to English constituency parsing (Results are on Section 23 of WSJ)

| Parser | Training | WSJ 23 F1 |
|---|---|---|
| Vinyals & Kaiser el al. (2014) [37] | WSJ only, discriminative | 88.3 |
| Petrov et al. (2006) [29] | WSJ only, discriminative | 90.4 |
| Zhu et al. (2013) [40] | WSJ only, discriminative | 90.4 |
| Dyer et al. (2016) [8] | WSJ only, discriminative | 91.7 |
| Transformer (4 layers) | WSJ only, discriminative | 91.3 |
| Zhu et al. (2013) [40] | semi-supervised | 91.3 |
| Huang & Harper (2009) [14] | semi-supervised | 91.3 |
| McClosky et al. (2006) [26] | semi-supervised | 92.1 |
| Vinyals & Kaiser el al. (2014) [37] | semi-supervised | 92.1 |
| Transformer (4 layers) | semi-supervised | 92.7 |
| Luong et al. (2015) [23] | multi-task | 93.0 |
| Dyer et al. (2016) [8] | generative | 93.3 |

increased the maximum output length to input length + 300. We used a beam size of 21 and $\alpha = 0.3$ for both WSJ only and the semi-supervised setting.

Our results in Table 4 show that despite the lack of task-specific tuning our model performs surprisingly well, yielding better results than all previously reported models with the exception of the Recurrent Neural Network Grammar [8].

In contrast to RNN sequence-to-sequence models [37], the Transformer outperforms the Berkeley-Parser [29] even when training only on the WSJ training set of 40K sentences.

## 7 Conclusion

In this work, we presented the Transformer, the first sequence transduction model based entirely on attention, replacing the recurrent layers most commonly used in encoder-decoder architectures with multi-headed self-attention.

For translation tasks, the Transformer can be trained significantly faster than architectures based on recurrent or convolutional layers. On both WMT 2014 English-to-German and WMT 2014 English-to-French translation tasks, we achieve a new state of the art. In the former task our best model outperforms even all previously reported ensembles.

We are excited about the future of attention-based models and plan to apply them to other tasks. We plan to extend the Transformer to problems involving input and output modalities other than text and to investigate local, restricted attention mechanisms to efficiently handle large inputs and outputs such as images, audio and video. Making generation less sequential is another research goals of ours.

The code we used to train and evaluate our models is available at `https://github.com/tensorflow/tensor2tensor`.

**Acknowledgements** We are grateful to Nal Kalchbrenner and Stephan Gouws for their fruitful comments, corrections and inspiration.

## References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[3] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V. Le. Massive exploration of neural machine translation architectures. *CoRR*, abs/1703.03906, 2017.

[4] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[6] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016.

[7] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[8] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars. In *Proc. of NAACL*, 2016.

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122v2*, 2017.

[10] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[12] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

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

[14] Zhongqiang Huang and Mary Harper. Self-training PCFG grammars with latent annotations across languages. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 832–841. ACL, August 2009.

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

[16] Łukasz Kaiser and Samy Bengio. Can active memory replace attention? In *Advances in Neural Information Processing Systems, (NIPS)*, 2016.

[17] Łukasz Kaiser and Ilya Sutskever. Neural GPUs learn algorithms. In *International Conference on Learning Representations (ICLR)*, 2016.

[18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099v2*, 2017.

[19] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. In *International Conference on Learning Representations*, 2017.

[20] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[21] Oleksii Kuchaiev and Boris Ginsburg. Factorization tricks for LSTM networks. *arXiv preprint arXiv:1703.10722*, 2017.

[22] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

[23] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015.

[24] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[25] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[26] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159. ACL, June 2006.

[27] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model. In *Empirical Methods in Natural Language Processing*, 2016.

[28] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

[29] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 433–440. ACL, July 2006.

[30] Ofir Press and Lior Wolf. Using the output embedding to improve language models. *arXiv preprint arXiv:1608.05859*, 2016.

[31] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[33] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[34] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2440–2448. Curran Associates, Inc., 2015.

[35] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[37] Vinyals & Kaiser, Koo, Petrov, Sutskever, and Hinton. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, 2015.

[38] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[39] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. Deep recurrent models with fast-forward connections for neural machine translation. *CoRR*, abs/1606.04199, 2016.

[40] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the ACL (Volume 1: Long Papers)*, pages 434–443. ACL, August 2013.

## Attention Visualizations

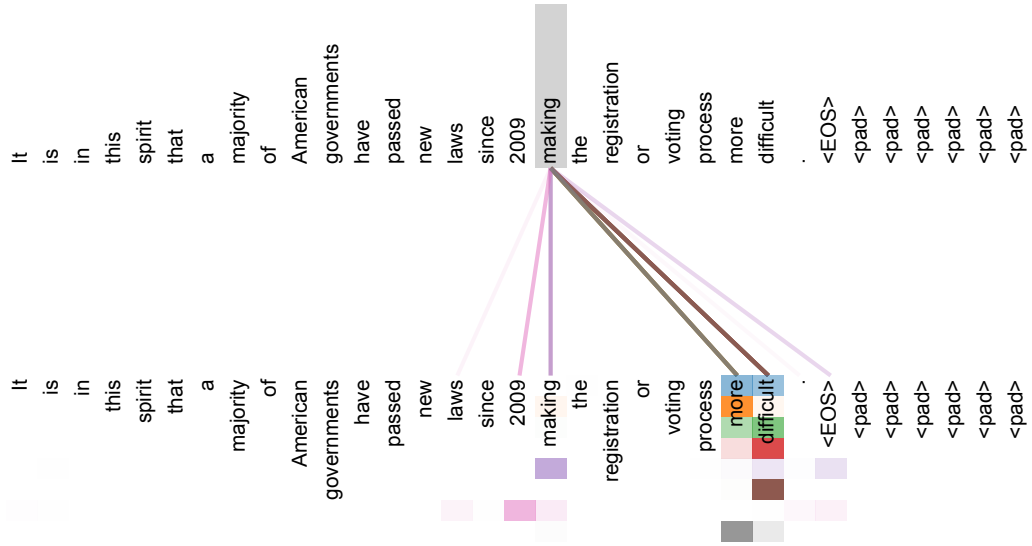


Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb 'making', completing the phrase 'making...more difficult'. Attentions here shown only for the word 'making'. Different colors represent different heads. Best viewed in color.

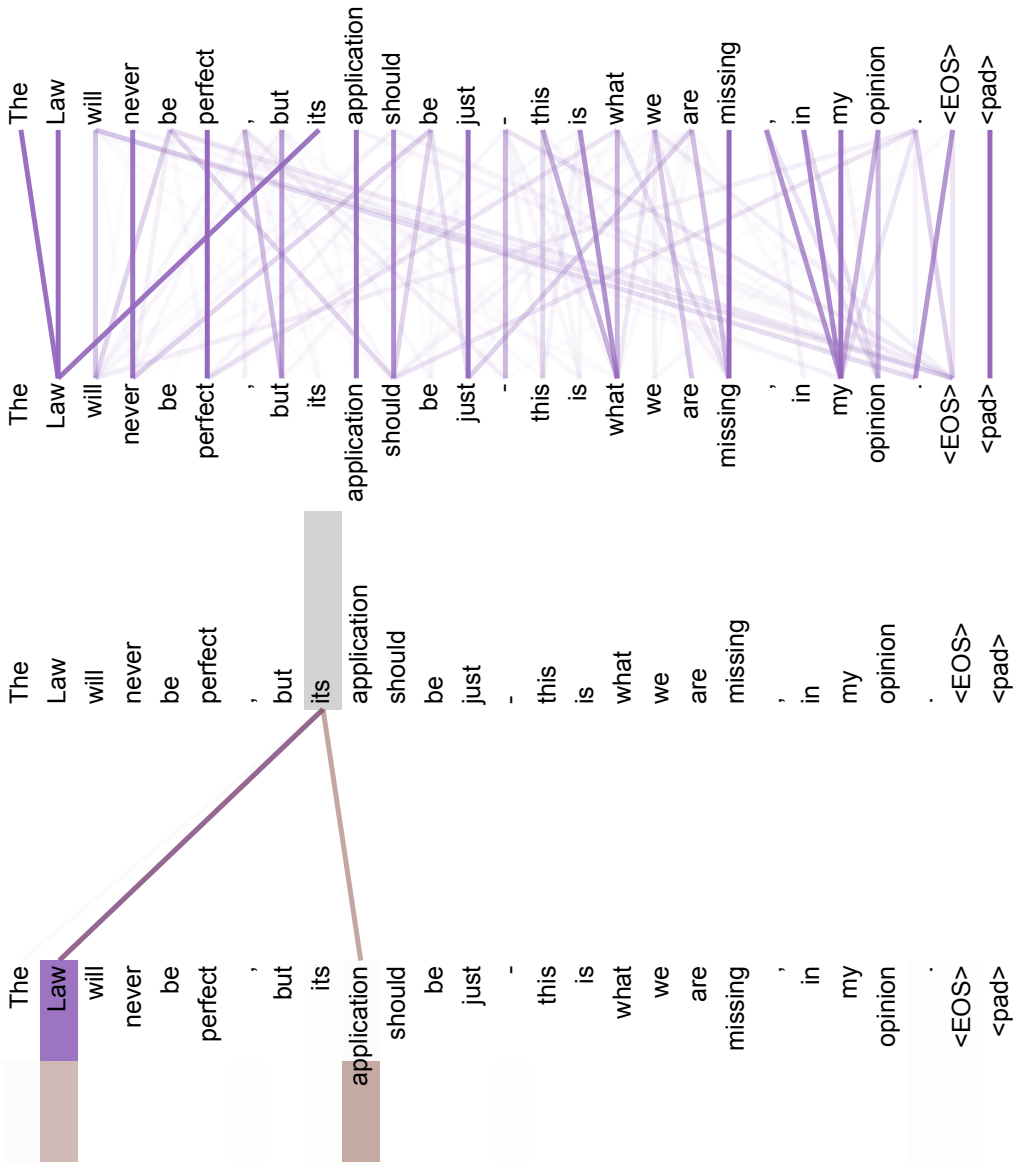


Figure 4: Two attention heads, also in layer 5 of 6, apparently involved in anaphora resolution. Top: Full attentions for head 5. Bottom: Isolated attentions from just the word 'its' for attention heads 5 and 6. Note that the attentions are very sharp for this word.

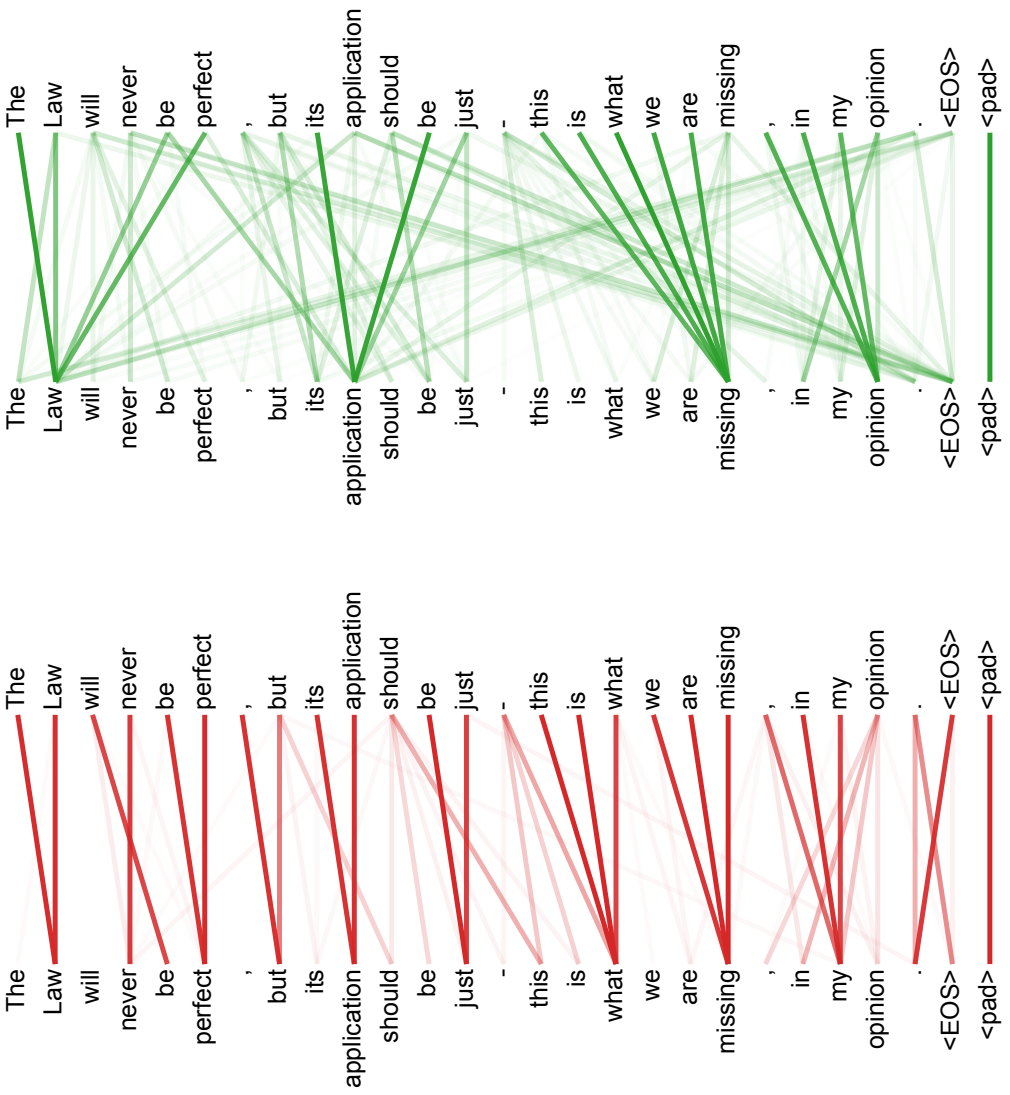


Figure 5: Many of the attention heads exhibit behaviour that seems related to the structure of the sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks.