SCHOOL OF COMPUTING INFORMATION, UNIVERSITY OF PITTSBURGH

Next-Day Mortality Prediction using Heterogeneous Data Sources

Anthony Sicilia, Muheng Yan, Sanchayan Sarkar

1 INTRODUCTION

While machine learning, and more specifically deep learning, has had a variety of application areas in recent years, perhaps one of the most impactful areas of application is medicine. For example, certain fully-convolutional neural networks [12] can be used for diverse segmentation tasks, automating time-expensive annotation tasks for radiologists. Beyond images, deep learning for time-series modeling has also been explored recently. A benchmark task [4] called **decompensation** was proposed. In this task, the goal of the model is to predict rapid deterioration of patient health in the next 24 hours. The importance of this problem is of course staggering. Providing medical practitioners with an early warning system for patient health deterioration would allow them intervene and possibly save lives. To this end, the goal of this work is to investigate the role of neural architectures in the task of decompensation in an ICU setting. Our focus is twofold

1. We aim to investigate recent advances in neural architectures for sequence-to-sequence modelling. In particular, we investigate the performance of an attention-based transformer module in comparison to the baseline LSTM.

2. We aim to investigate the role of multiple heterogeneous data sources. We present an embedding-based approach for modelling and aggregating a time-series of events from each source. Further, we conduct leave-one-out experiments to determine the usefulness of each data source.

The structure of this work is the following: in Section 2 we cover some preliminaries surrounding the task of decomponsentation and the related task of sequential modelling, in Section 3 we formally outline our proposed approach, in Section 4 we conduct experimental validation

of our models with detailed analysis of strengths, weakness, and contribution of different data sources, and finally in Section 5 we conclude our work.

2 Related Works

Decompensation As mentioned, [4] propose the task of decompensation. The aim of this task is to predict *at each hour* the deterioration of the patient's health in the next two hours. The model processes input sequentially, making predictions at each hour (similar to an *in-the-wild* deployment). The authors propose a recurrent neural architecture known as Long Short Term Memory (LSTM) for this problem, further investigating modifications to the learning curriculum and architecture (e.g. a multi-task loss, bidirectional processing, etc.).

Our approach differs in two ways. First, our prediction task is more coarse – we make predictions at daily rather than hourly rate. Second, our focus is on investigating the benefit of recent attention-based models over the traditional LSTM. Third, we investigate a wider variety of data sources. While our approach is certainly not directly comparable to the work done in [4] due to differences in time-scale, differences in available information, and differences in setup, we show their results in Section 4.3 as a point of reference.

Sequential Modelling We inspect the next-day prediction task as a sequential modeling problem. Specifically, we are building sequential model with neural networks. The line of works in modeling sequential data with neural network contains three general approaches: convolutional neural networks (CNNs), recurrent neural networks (RNNs), and attention-based models.

Convolutional neural networks [6] is first proposed in modeling images. It created convolution kernels that group up locally close elements (pixels in images, or steps in sequences) to aggregate them into unified latent representations. Though CNN is capable of modelling sequences, it has been shown that other models usually outperform CNN in an abundant of sequential modeling applications including NLP and sequential forecasting. For this reason, in our work we do not include CNN as one of our approaches.

Recurrent neural network [7] creates a serialized perceptrons with the respect to the data time steps. Each cell takes not only the time step input but the hidden state output from the previous perceptron as well. In this way the network encodes the sequential dependency in the output of the final step. There exist variant versions of RNN such as GRU [2] and LSTM [5], each addressing limitations of the simple RNN.

Attention-based model [13] is a recent alternative architecture to RNNs. Instead of encoding the sequential data step by step, the attention mechanism soft select relevant steps within a sequence regardless of the distance. Compared to RNNs, attention-based layers is more powerful finding long term dependencies and non-sequential dependencies. It is shown that attention models outperform RNNs in many tasks [1, 11].

In our work we leverage both the RNNs and the attention-based models to model the time series medical record data. We illustrate the details of models in the following section and propose a comprehensive comparison on their performances in the prediction.

3 METHOD

In this section, we describe our assumed framework in a general sense. For more details on the particular data we consider, we direct the reader to Section 4.1. Recall, the task we are interested in is prediction of next-day mortality of admissions at each time-point during their stay. We describe the notation for a particular patient of interest. At the label level, we assume a binary sequence $(y_t)_{t=1}^T$ indicating whether the patient died t days after admission. At the input level, we assume s data sources, each containing a pre-prescribed list of possible events. For example, in our task, a data source may come from lab results where an *abnormal* test result would correspond to an event we would like to track. For each data source, and a particular patient, we assume we are given a sequence of sets $\left(\mathcal{X}_t^s\right)_{t=1}^T$ where $\mathcal{X}_t^s \subset \{0,1\}^{N_d+1}$ contains the one-hot indicator vectors for the N_s possible events in data source s (including an additional `NULL` to indicate that no event occurred at time point t). In particular, if $\mathbf{x} \in \mathcal{X}_t^s$, then $||\mathbf{x}|| = 1$ and $\mathbf{x}_i = 1$ implies event i of data source s occurred during day t.

3.1 MODELING EVENTS

For a particular event $\mathbf{x} \in \mathcal{X}_t^s$, we wish to model this event with a semantically meaningful vector; i.e., allowing capture of information to be used later in the modeling pipeline. While there are many possible ways to achieve this (e.g. using SVD, LDA, or Skip-Gram [8] to model co-occurrence between events and admissions), we choose to let our representations be learned based on task feedback. That is, we specify learnable vector representations for each event and update these representations based on back-propagation from the task loss. If we group the event representations for data source s so that they form the columns of a weight matrix $\mathbf{W}^s$, then we may extract the vector representation of event $\mathbf{x} \in \mathcal{X}_t^s$ with the following matrix-vector multiplication

$$\mathbf{W}^s\mathbf{x} \tag{3.1}$$

For a particular day t and data source s, we have the entire set of events $\mathcal{X}_t^s$. To consolidate the information from all the events into a single vector $\mathbf{e}_t^s$, we simply compute an average as below

$$\mathbf{e}_t^s = \frac{1}{|\mathcal{X}_t^d|} \sum_{\mathbf{x} \in \mathcal{X}_t^d} \mathbf{W}^s\mathbf{x} \tag{3.2}$$

Further, we would like one vector representation $\mathbf{e}_t$ for each day across our s data sources. This is easily accomplished by concatenation. In particular, if each for each s we have $\mathbf{e}_t^s \in \mathbb{R}^m$ then after concatenation we have $\mathbf{e}_t \in \mathbb{R}^{ms}$.

3.2 TIME SEQUENCE MODELING

Our task is next-day mortality prediction and we frame this task as an sequence-to-sequence problem. In particular, we would like translate the sequence $(\mathbf{e}_t)_{t=1}^{T-1}$ to sequence $(y_t)_{t=2}^T$. We emphasize that the sequences are shifted. This is in accordance with our *next*-day prediction task. In particular, this shift ensures that our model would be useful to a practitioner because it uses no information from the day of mortality to make its predictions. This means that no *obvious* indicators such as discharge summaries can lead to inflated model performance and

also means that our model would serve as an *early* warning system ensuring practitioners have time to intervene.

We frame as solution to this task a number of neural modules. In particular, we assume a neural module g with parameters θ which is a function mapping from our sequence $(\mathbf{e}_t)_{t=1}^{T-1}$ to a hidden state $(\mathbf{h}_t)_{t=1}^{T-1}$; i.e.

$$g\left((\mathbf{e}_t)_{t=1}^{T-1};\theta\right) = (\mathbf{h}_t)_{t=1}^{T-1} \tag{3.3}$$

We map this hidden representation to logits for our task with a weight vectors $\mathbf{u}_0$ and $\mathbf{u}_1$, so that we can extract probabilities as below

$$p(y_t = 1|\mathbf{h}_{t-1}) = \frac{e^{\mathbf{u}_1\mathbf{h}_{t-1}}}{e^{\mathbf{u}_1\mathbf{h}_{t-1}} + e^{\mathbf{u}_0\mathbf{h}_{t-1}}} \tag{3.4}$$

Below we describe two possible choices for neural modules to extract the hidden states $(\mathbf{h}_t)_{t=1}^{T-1}$.

3.3 Encoder Modules

We focus on applying the LSTM or Long-Short Term Memory network [5] which processes input sequentially and outputs a hidden state at each time step and the Transformer module [] which translates one sequence to another via an attention mechanism [1]. We describe the modules in details below.

LSTM is a recurrent neural network variant which processes input sequentially generating a hidden state at each time step. We use this hidden state to construct the sequence $(\mathbf{h}_t)_{t=1}^{T-1}$ employed above. In particular, for our implementation [10], using the same notation as above, the processing steps of the LSTM can be written as below

$$\mathbf{i}_t = \sigma(\mathbf{b}^{\prime} + \mathbf{U}^{\prime}\mathbf{e}_t + \mathbf{W}^{\prime}\mathbf{h}_{(t-1)}) \tag{3.5}$$

$$\mathbf{f}_t = \sigma(\mathbf{b}^{f} + \mathbf{U}^{f}\mathbf{e}_t + \mathbf{W}^{f}\mathbf{h}_{(t-1)}) \tag{3.6}$$

$$\mathbf{c}_t = \mathbf{f}_t \times \mathbf{c}_{(t-1)} + \mathbf{i}_t \times \tanh(\mathbf{b}^{c} + \mathbf{U}^{c}\mathbf{e}_t + \mathbf{W}^{c}\mathbf{h}_{(t-1)}) \tag{3.7}$$

$$\mathbf{o}_t = \sigma(\mathbf{b}^{o} + \mathbf{U}^{o}\mathbf{e}_t + \mathbf{W}^{o}\mathbf{h}_{(t-1)}) \tag{3.8}$$

$$\mathbf{h}_t = \mathbf{o}_t \times \tanh(\mathbf{c}_t) \tag{3.9}$$

where each $\mathbf{b}^*, \mathbf{U}^*, \mathbf{W}^*$ are learned, activation functions are applied point-wise, and $\times$ indicates an element-wise product. Conceptually these equations amount to an input-gate, a forget-gate, and an output gate (given by the sigmoid activations) which modulate the flow of information; in particular, they help solve issues of vanishing gradients and allow retained memory in the cell-state for handling of longer input-sequences [3]. In our application, we usually employ a stacked LSTM [9] which feeds the hidden state output by one LSTM into a subsequent LSTM to achieve a multi-layer structure capable of better feature representation.

[1] Therefore, we pad sequence with zeros since the attention mechanism requires fixed length input

TRANSFORMER is a fully-connected attention-based model architecture [13]. It is an alternative model for modeling sequences to recurrent neural networks that first proposed for natural language processing tasks. Recent works show that it has a better performance compared to RNNs [11, 1].

Using the notation above, we stack our sequence of event embeddings $(\mathbf{e}_t)_{t=1}^{T-1}$ to form the matrix $\mathbf{X} \in \mathbb{R}^{(T-1)\times d}$. Mapping from a particular embedding $\mathbf{e}_t$ to its hidden state $\mathbf{h}_t$ can then be described as follows. First, the transformer encoder maps a sequence of vectors $\mathbf{X}$ into a query vector $\mathbf{q} \in \mathbb{R}^{1\times d}$, a key vector and a query vector $\mathbf{k}, \mathbf{v} \in \mathbb{R}^{1\times d}$ to soft select the relevant steps in sequence with *scaled dot-product attention* (SDP Attention):

$$\text{Attn}(\mathbf{q}, \mathbf{k}, \mathbf{v}) = \text{softmax}(\frac{\mathbf{q}\mathbf{k}^T}{\sqrt{d}})\mathbf{v} \tag{3.10}$$

Here, the mappings from $\mathbf{X}$ to $\mathbf{q}, \mathbf{k}$ and $\mathbf{v}$ are given by learnable parameters. Stacking multiple SDP attentions we may have Multi-Head attentions as:

$$\text{Multi-Head}(\mathbf{X}) = \text{concatenation}(\mathbf{a}_1, ..., \mathbf{a}_k)\mathbf{W}^O \tag{3.11}$$

where $\mathbf{a}_i$ is the output of the i^{th} SDP Attention, and $\mathbf{W}^O$ is a learnable mapping matrix that maps the concatenated heads back to dimension d (if it is not the last layer) or the desired size of $\mathbf{h}_t$ (if it is the last layer). As implied, stacking multiple layers of the Multi-Head attentions we may construct our full transformer encoder. The sequence of latent vectors output by the transformer encoder is used to define the hidden states $(\mathbf{h}_t)_{t=1}^{T-1}$ defined above.

4 EXPERIMENT

We report our experiment and results in this section. The two different types of models reported in the previous section are implemented and applied to MIMIC-III the Electronic Health Records (EHR) dataset. We aim to predict the mortality of patients on a daily basis. We will first describe the dataset in Section 4.1, our experiment settings in Section 4.2, and report the model performances on the prediction in Section 4.3. We also compare the two model choices and discuss the predictive features in the dataset in Section 4.4

4.1 DATASET

MIMIC-III dataset contains the medical records of 19,837 patients admitted to ICUs of Beth Israel Deaconess Medical Center **??**. The patient samples are split into three sets of train, dev and test for training, validating, and evaluating the models respectively. Table 4.1 reports the sizes of all three sets. Besides a time-series flag indicating the mortality of the patients, each admission record of the patients contains a series of events, test results, and observations. These events are all having a timestamp and are categorized by their arch-types as follows:

LAB EVENTS reflect the lab test operated to the patients. Each lab events has a unique ID, associated with the specific tests and the readings of them.

MEDICATIONS are the medications provides to the patients. Each is recorded with the type of the medication and their dosages.

CHART EVENTS records the vital signs of the patients. The readings include blood pressure, heart rate, respiratory rates, body temperatures and others.

MICROBIOLOGY contains the information that a foreign organism infection. The infection is labeled with the sample where founded (e.g. blood).

TEXT RECORDS are the records written by the doctors. Each text record is noted with a type label, for instance, discharge notes, progress report, etc.

	Train	Dev	Test	Total
#Admissions	13,552 (68.3%)	1,935 (9.8%)	4,350 (21.9%)	19,837

Table 4.1: Distribution of the Samples in Sets

4.2 EXPERIMENT SETTINGS

We code each type of the events and observations as one source of the input data by the method reported in Section 3.1. We set up the embeddings for all event types with the size 32, and input them into our encoder models. The LSTM encoder consists of two RNN layes, with hidden size 256 and dropout rate 0.3. The Transformer model has two layers of the multi-head attention, each with 4 heads of 256 hidden size and 0.1 dropout rate. Both encoders are connected to one dense layer with output size 2, for the reason that we are predicting the binary outcome of die/survive.

The model is optimized with stochastic gradient descent optimizer with momentum and initial learning rate at 0.01. We run 5 learning epochs for the models, and validate and evaluate them each 5,000 steps. We use two metrics to evaluate the models: Area Under Curve of the Receiver Operating Characteristic Curve (AUC-ROC) and Area Under Curve of the Precision-Recall Curve (AUC-PRC). These metrics are shown to be more robust than others when the ground-truth label distribution is highly imbalanced.

4.3 EXPERIMENT RESULTS

We show our main results in Table 4.2. For the transformer, we show the model trained on all data sources as well as with each data source omitted. For LSTM, we show only the model trained on all data sources since this was best. Comparing the best performing variant of our models (LSTM or transformer), we see that the transformer is better in terms of both AUC-ROC and AUC-PR. In particular, the transformer is able to more than double the performance of the best LSTM when measuring AUC-PR. This is important as AUC-PR is a good measure to compare when the label space is imbalanced (as in our problem). Further, in comparison to some relevant benchmarks [4], we see that our model is performing comparably. While these

benchmarks test on the same dataset, we refrain from making a direct comparison with these methods since our experimental setups are different, and more importantly, [4] make hourly predictions rather than daily predictions which can effect the results.

Models	AUC-ROC	AUC-PR
LSTM	0.878	0.157
Transformer	**0.91**	**0.355**
Benchmarks*		
LR	0.870	0.214
C	0.906	0.333
C + DS	0.911	0.344

Excluded Data Source	AUC-ROC	AUC-PR
Note Event Types	0.882	0.346
MicroBio Events	**0.910**	**0.355**
Medication Events	0.855	0.189
Chart Events	0.891	0.327
Lab Events	0.868	0.315
All Data Source	0.850	0.327

* The Benchmark model performances are reported by Harutyunyan et al. [4]. These models are predicting mortality on a hourly time series and thus are not directly comparable to ours.

Table 4.2: **LEFT** Between-model comparison of the sequential mortality prediction performances. For our models, *best* performing for each variant is shown. **RIGHT** The Transformer performances with the specific data source excluded.

4.4 Model Discussion

In this section, we compare and contrast the performance of the best LSTM and best Transformer model to understand there strengths and weaknesses. We refer the reader to the Table of Figures 4.3 for a visual accompaniment to each discussion point.

ROC-Curves indicate a model's ability to discern between the positive and negative classes by evaluating the FPR and TPR at a variety of thresholds. The area under the ROC-Curve is a good measure of this (as was shown in Table 4.2). While from previous results it is clear the transformer performs better under this metric, we may use the actual ROC-Curve to discuss why. In particular, we can see that the transformer maintains for thresholds which produce a low FPR in the ranges [0.0, 0.2], the transformer is able to produce a high TPR while the LSTM is not. Therfore, if the medical practitioner were willing to sacrifice FPR at the expense of TPR, our experiments indicate the transformer is an obvious choice for maximizing this metric.

PR-Curves indicate a model's ability to discern between the positive and negative classes by evaluating the Precision and Recall at a variety of thresholds. It is particularly useful in the case of imbalanced data (as visible, it is a harder metric to maximize in these cases). Again, Table 4.2 indicates that the transformer performs better under this metric. By examining the curves, we can see that this is uniformly true for most values of Precision and Recall.

Calibration Curves provide a useful visual of the models confidence in terms of its probability estimates. To produce the curve, model predictions are first binned. For each bin, the model's average predicted probability is computed (shown on the horizontal axis) and the *true* fraction of positives in this bin is computed (shown on the vertical axis). In this way, one

can visualize whether a bin's *true* probability is close to the model's predicted probability. By examining the curves, we can see that the transformer is generally more conservative than the LSTM. Since the transformer's curve floats above the line which corresponds to *perfect* prediction, we can see that it is generally under-confident, providing predicted probabilities lower than reality. The transformer is also conservative in terms of the range of its predictions, rarely predicting a probability higher than 0.4. In contrast, the LSTM is generally over-confident, sometimes predicting probabilities near 1 when in reality, this bin reflects a probability closer to 0.4.

4.5 Measuring Data Source Impact

In this section, we discuss the impact of the different data sources. To gain data on this, we perform leave one out experiments on each data source; i.e., we train the model excluding the data source. We refer the reader to Table 4.2 for the results of these. While most changes are somewhat unremarkable, we notice two data sources which can have a significant (or at least unexpected) impact on performance. First, there is a significant drop in performance when omitting medication events. This seems reasonable as the amount and kinds of medical interventions can be indicative of declining patient status; e.g., a patient undergoing renal failure or sepsis would have a number of typically associated medications added to their input list. On this point though, we do remark that the goal of the model is to warn the medical practitioner and as a result *motivate* such medical interventions, so this raises two points of concern: (i) the model is warning an already aware medical practitioner and (ii) the model may experience a detrimental distributional shift during an *in the wild* deployment.

Lastly, we note that omitting micro-organisms improves the model performance. This could be because positive micro-organisms identification is a noisy data source without much predictive power; i.e, removing the data source attenuates the model to relevant data source.

5 Discussion

In this work, we investigate the efficacy of two neural architectures in predicting next-day mortality based on heterogeneous data. We show that the transformer may be better suited at this task than traditional RNNs such as the LSTM. We further provide a detailed discussion on strengths and weaknesses of both models. Finally, we investigate the contribution of individual data sources, finding that medications are a key indicator. While we show promising results, the import task of next-day mortality prediction is far from solved. While high class imbalance makes the problem challenging as it is, *in-the-wild* deployment of such models poses even more challenges such as measuring and handling distribution-shift and data-uncertainty.

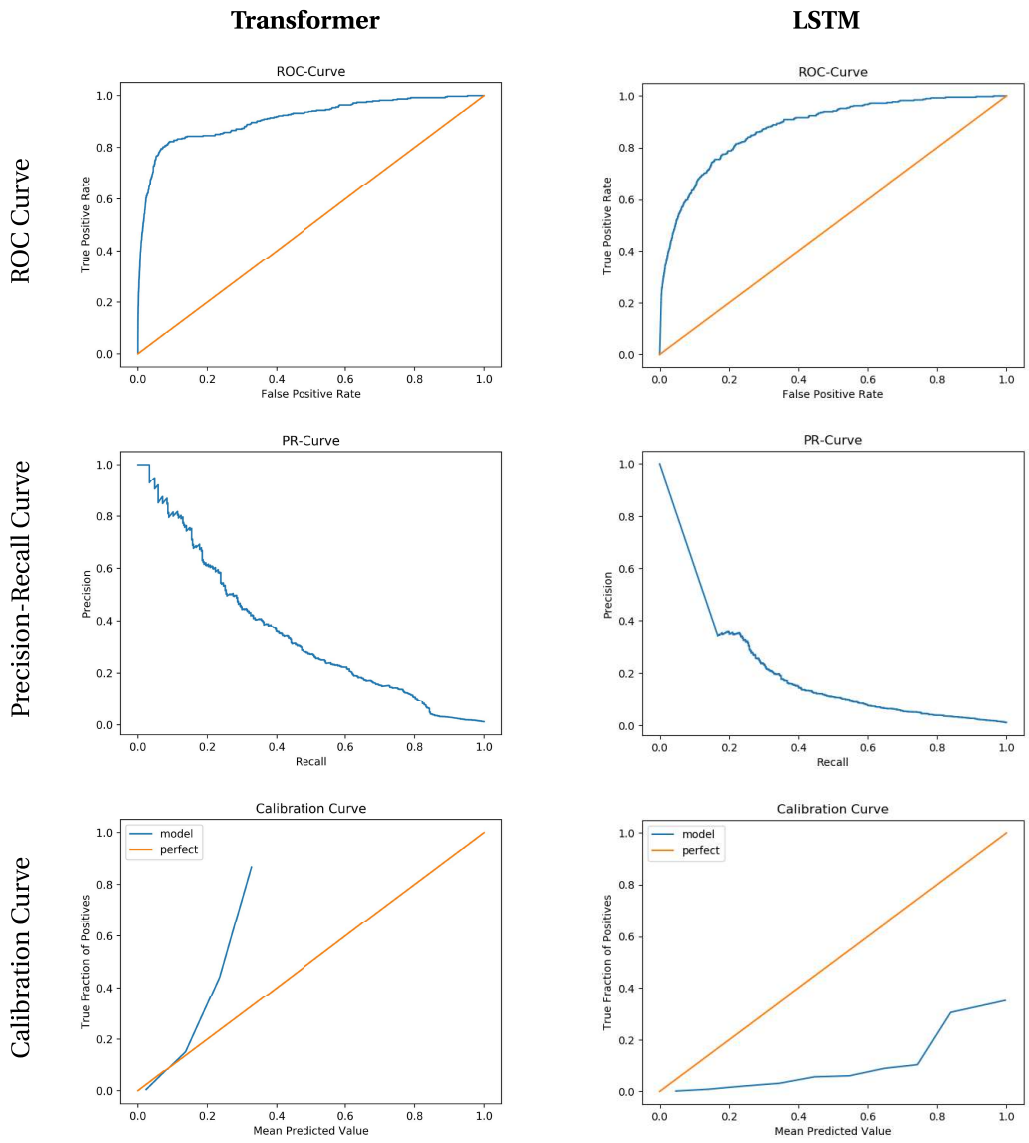

Table 4.3: Receiver Operating Characteristic (ROC) curves, Precision-Recall Curves, and Calibration Curves of the best LSTM and Transformer model.

REFERENCES

[1] 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.

[2] Rahul Dey and Fathi M Salemt. Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pages 1597–1600. IEEE, 2017.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[4] Hrayr Harutyunyan, Hrant Khachatrian, David C Kale, Greg Ver Steeg, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *arXiv preprint arXiv:1703.07771*, 2017.

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

[6] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[7] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[8] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[9] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013.

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[11] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

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