

Pyramidal Recurrent Unit for Language Modeling

Sachin Mehta¹, Rik Koncel-Kedziorski¹, Mohammad Rastegari², and Hannaneh Hajishirzi¹

¹University of Washington, Seattle, WA, USA

{sacmehta, kedzior, hannaneh}@uw.edu

²Allen Institute for AI and XNOR.AI, Seattle, WA, USA

mohammadr@allenai.org

Abstract

LSTMs are powerful tools for modeling contextual information, as evidenced by their success at the task of language modeling. However, modeling contexts in very high dimensional space can lead to poor generalizability. We introduce the Pyramidal Recurrent Unit (PRU), which enables learning representations in high dimensional space with more generalization power and fewer parameters. PRUs replace the linear transformation in LSTMs with more sophisticated interactions including pyramidal and grouped linear transformations. This architecture gives strong results on word-level language modeling while reducing the number of parameters significantly. In particular, PRU improves the perplexity of a recent state-of-the-art language model Merity et al. (2018) by up to 1.3 points while learning 15-20% fewer parameters. For similar number of model parameters, PRU outperforms all previous RNN models that exploit different gating mechanisms and transformations. We provide a detailed examination of the PRU and its behavior on the language modeling tasks. Our code is open-source and available at <https://sacmehta.github.io/PRU/>.

1 Introduction

Long short term memory (LSTM) units (Hochreiter and Schmidhuber, 1997) are popular for many sequence modeling tasks and are used extensively in language modeling. A key to their success is their articulated gating structure, which allows for more control over the information passed along the recurrence. However, despite the sophistication of the gating mechanisms employed in LSTMs and similar recurrent units, the input and context vectors are treated with simple linear transformations prior to gating. Non-linear transformations such as convolutions (Kim et al., 2016) have been used, but these have not achieved the

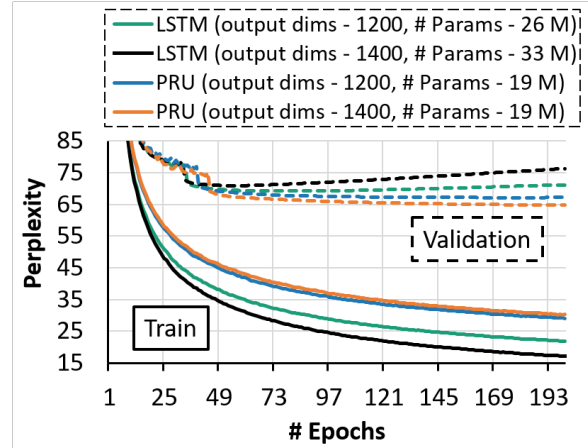


Figure 1: Comparison of training (solid lines) and validation (dashed lines) perplexities on the Penn Treebank with standard dropout for pyramidal recurrent units (PRU) and LSTM. PRUs learn latent representations in very high-dimensional space with good generalizability and fewer parameters. See Section 3 for more details about PRUs. Best viewed in color.

performance of well regularized LSTMs for language modeling (Melis et al., 2018).

A natural way to improve the expressiveness of linear transformations is to increase the number of dimensions of the input and context vectors, but this comes with a significant increase in the number of parameters which may limit generalizability. An example is shown in Figure 1, where LSTMs performance decreases with the increase in dimensions of the input and context vectors. Moreover, the semantics of the input and context vectors are different, suggesting that each may benefit from specialized treatment.

Guided by these insights, we introduce a new recurrent unit, the Pyramidal Recurrent Unit (PRU), which is based on the LSTM gating structure. Figure 2 provides an overview of the PRU. At

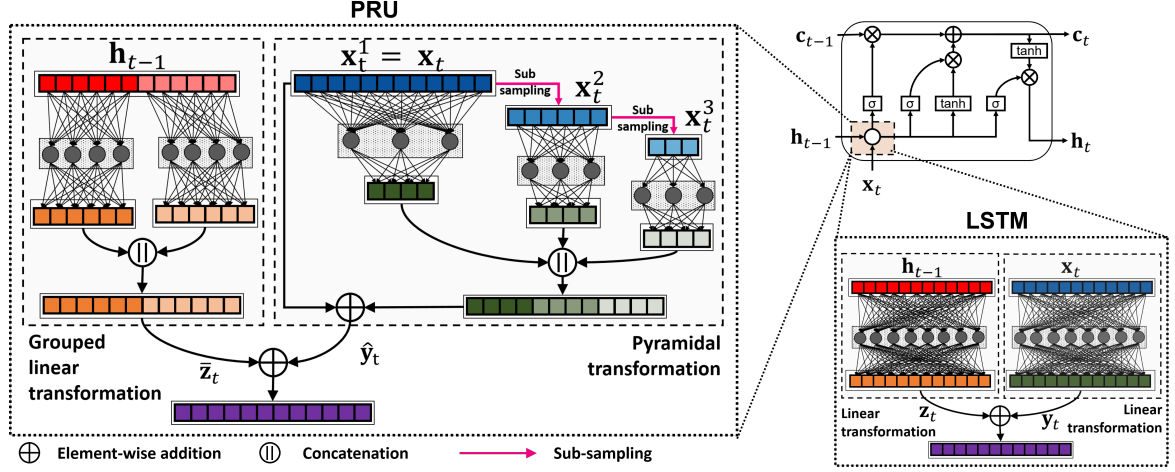


Figure 2: Block diagram visualizing the transformations in pyramidal recurrent unit (left) and the LSTM (bottom right) along with the LSTM gating architecture (top right). Blue, red, green (or orange), and purple signify the current input x_t , output of the previous cell h_{t-1} , the output of transformations, and the fused output, respectively. The color intensity is used to represent sub-sampling and grouping operations.

the heart of the PRU is the *pyramidal transformation* (PT), which uses subsampling to effect multiple views of the input vector. The subsampled representations are combined in a pyramidal fusion structure, resulting in richer interactions between the individual dimensions of the input vector than is possible with a linear transformation. Context vectors, which have already undergone this transformation in the previous cell, are modified with a *grouped linear transformation* (GLT) which allows the network to learn latent representations in high dimensional space with fewer parameters and better generalizability (see Figure 1).

We show that PRUs can better model contextual information and demonstrate performance gains on the task of language modeling. The PRU improves the perplexity of the current state-of-the-art language model (Merity et al., 2018) by up to 1.3 points, reaching perplexities of 56.56 and 64.53 on the Penn Treebank and WikiText2 datasets while learning 15-20% fewer parameters. Replacing an LSTM with a PRU results in improvements in perplexity across a variety of experimental settings. We provide detailed ablations which motivate the design of the PRU architecture, as well as detailed analysis of the effect of the PRU on other components of the language model.

2 Related work

Multiple methods, including a variety of gating structures and transformations, have been pro-

posed to improve the performance of recurrent neural networks (RNNs). We first describe these approaches and then provide an overview of recent work in language modeling.

Gating-based mechanisms: The performance of RNNs have been greatly improved by gating mechanisms such as LSTMs (Hochreiter and Schmidhuber, 1997), GRUs (Chung et al., 2014), peep-hole connections (Gers and Schmidhuber, 2000), SRUs (Lei et al., 2018), and RANs (Lee et al., 2017). In this paper, we extend the gating architecture of LSTMs (Hochreiter and Schmidhuber, 1997), a widely used recurrent unit across different domains.

Transformations: Apart from the widely used linear transformation for modeling the temporal data, another transformation that has gained popularity is convolution (LeCun et al., 1995). Convolution-based methods have gained attention in computer vision tasks (Krizhevsky et al., 2012) as well as some of the natural language processing tasks including machine translation (Gehring et al., 2017). Convolution-based methods for language modeling, such as CharCNN (Kim et al., 2016), have not yet achieved the performance of well regularized LSTMs (Melis et al., 2018). We inherit ideas from convolution-based approaches, such as sub-sampling, to learn richer representations (Krizhevsky et al., 2012; Han et al., 2017).

Regularization: Methods such as dropout (Srivastava et al., 2014), variational dropout (Kingma et al., 2015), and weight dropout (Merity et al., 2018) have been proposed to regularize RNNs. These methods can be easily applied to PRUs.

Other efficient RNN networks: Recently, there has been an effort to improve the efficiency of RNNs. These approaches include quantization (Xu et al., 2018), skimming (Seo et al., 2018; Yu et al., 2017), skipping (Campos et al., 2018), and query reduction (Seo et al., 2017). These approaches extend standard RNNs and therefore, these approaches are complementary to our work.

Language modeling: Language modeling is a fundamental task for NLP and has garnered significant attention in recent years (see Table 1 for comparison with state-of-the-art methods). Merity et al. (2018) introduce regularization techniques such as weight dropping which, coupled with a non-monotonically triggered ASGD optimization, achieves strong performance improvements. Yang et al. (2018) extend Merity et al. (2018) with the mixture of softmaxes (MoS) technique, which increases the rank of the matrix used to compute next-token probabilities. Further, Merity et al. (2017) and Krause et al. (2018) propose methods to improve inference by adapting models to recent sequence history. Our work is complementary to these recent softmax layer and inference procedure improvements.

We extend state-of-the-art language model in Merity et al. (2018) by replacing the LSTM with the PRU. We show by experiments that the PRU improves the performance of Merity et al. (2018) while learning fewer parameters.

3 Pyramidal Recurrent Units

We introduce Pyramidal Recurrent Units (PRUs), a new RNN architecture which improves modeling of context by allowing for higher dimensional vector representations while learning fewer parameters. Figure 2 provides an overview of PRU. We first elaborate on the details of the pyramidal transformation and the grouped linear transformation. We then describe our recurrent unit, PRU.

3.1 Pyramidal transformation for input

The basic transformation in many recurrent units is a linear transformation \mathcal{F}_L defined as:

$$\mathbf{y} = \mathcal{F}_L(\mathbf{x}) = \mathbf{W} \cdot \mathbf{x}, \quad (1)$$

where $\mathbf{W} \in \mathbb{R}^{N \times M}$ are learned weights that linearly map $\mathbf{x} \in \mathbb{R}^N$ to $\mathbf{y} \in \mathbb{R}^M$. To simplify notation, we omit the biases.

Motivated by successful applications of sub-sampling in computer vision (e.g., (Burt and Adelson, 1987; Lowe, 1999; Krizhevsky et al., 2012; Mehta et al., 2018)), we subsample input vector \mathbf{x} into K pyramidal levels to achieve representation of the input vector at multiple scales. This sub-sampling operation produces K vectors, represented as $\mathbf{x}^k \in \mathbb{R}^{\frac{N}{2^{k-1}}}$, where 2^{k-1} is the sampling rate and $k = \{1, \dots, K\}$. We learn scale-specific transformations $\mathbf{W}^k \in \mathbb{R}^{\frac{N}{2^{k-1}} \times \frac{M}{K}}$ for each $k = \{1, \dots, K\}$. The transformed subsamples are concatenated to produce the pyramidal analog of \mathbf{y} , here denoted as $\bar{\mathbf{y}} \in \mathbb{R}^M$:

$$\bar{\mathbf{y}} = \mathcal{F}_P(\mathbf{x}) = [\mathbf{W}^1 \cdot \mathbf{x}^1, \dots, \mathbf{W}^K \cdot \mathbf{x}^K], \quad (2)$$

where $[\cdot, \cdot]$ indicates concatenation. We note that pyramidal transformation with $K = 1$ is the same as the linear transformation.

To improve gradient flow inside the recurrent unit, we combine the input and output using an element-wise sum (when dimension matches) to produce residual analog of pyramidal transformation, as shown in Figure 2 (He et al., 2016).

Sub-sampling: We sub-sample the input vector \mathbf{x} into K pyramidal levels using the kernel-based approach (LeCun et al., 1995; Krizhevsky et al., 2012). Let us assume that we have a kernel κ with $2e + 1$ elements. Then, the input vector \mathbf{x} can be sub-sampled as:

$$\mathbf{x}^k = \sum_{i=1}^{N/s} \sum_{j=-e}^e \mathbf{x}^{k-1}[si]\kappa[j], \quad (3)$$

where s represents the stride and $k = \{2, \dots, K\}$.

Reduction in parameters: The number of parameters learned by the linear transformation and the pyramidal transformation with K pyramidal levels to map $\mathbf{x} \in \mathbb{R}^N$ to $\bar{\mathbf{y}} \in \mathbb{R}^M$ are NM and $\frac{NM}{K} \sum_{k=1}^K 2^{(1-k)}$ respectively. Thus, pyramidal transformation reduces the parameters of a linear transformation by a factor of $K(\sum_{k=1}^K 2^{(1-k)})^{-1}$. For example, the pyramidal transformation (with $K = 4$ and $N = M = 600$) learns 53% fewer parameters than the linear transformation.

3.2 Grouped linear transformation for context

Many RNN architectures apply linear transformations to both the input and context vector. However, this may not be ideal due to the differing semantics of each vector. In many NLP applications including language modeling, the input vector is a dense word embedding which is shared across all contexts for a given word in a dataset. In contrast, the context vector is highly contextualized by the current sequence. The differences between the input and context vector motivate their separate treatment in the PRU architecture.

The weights learned using the linear transformation (Eq. 1) are reused over multiple time steps, which makes them prone to over-fitting (Gal and Ghahramani, 2016). To combat over-fitting, various methods, such as variational dropout (Gal and Ghahramani, 2016) and weight dropout (Merity et al., 2018), have been proposed to regularize these recurrent connections. To further improve generalization abilities while simultaneously enabling the recurrent unit to learn representations at very high dimensional space, we propose to use grouped linear transformation (GLT) instead of standard linear transformation for recurrent connections (Kuchaiev and Ginsburg, 2017). While pyramidal and linear transformations can be applied to transform context vectors, our experimental results in Section 4.4 suggests that GLTs are more effective.

The linear transformation $\mathcal{F}_L : \mathbb{R}^N \rightarrow \mathbb{R}^M$ maps $\mathbf{h} \in \mathbb{R}^N$ linearly to $\mathbf{z} \in \mathbb{R}^M$. Grouped linear transformations break the linear interactions by factoring the linear transformation into two steps. First, a GLT splits the input vector $\mathbf{h} \in \mathbb{R}^N$ into g smaller groups such that $\mathbf{h} = \{\mathbf{h}^1, \dots, \mathbf{h}^g\}, \forall \mathbf{h}^i \in \mathbb{R}^{\frac{N}{g}}$. Second, a linear transformation $\mathcal{F}_L : \mathbb{R}^{\frac{N}{g}} \rightarrow \mathbb{R}^{\frac{M}{g}}$ is applied to map \mathbf{h}^i linearly to $\mathbf{z}^i \in \mathbb{R}^{\frac{M}{g}}$, for each $i = \{1, \dots, g\}$. The g resultant output vectors \mathbf{z}^i are concatenated to produce the final output vector $\bar{\mathbf{z}} \in \mathbb{R}^M$.

$$\bar{\mathbf{z}} = \mathcal{F}_G(\mathbf{h}) = [\mathbf{W}^1 \cdot \mathbf{h}^1, \dots, \mathbf{W}^g \cdot \mathbf{h}^g] \quad (4)$$

GLTs learn representations at low dimensionality. Therefore, a GLT requires g fewer parameters than the linear transformation. We note that GLTs are subset of linear transformations. In a linear transformation, each neuron receives an input from each element in the input vector while in a

GLT, each neuron receives an input from a subset of the input vector. Therefore, GLT is the same as a linear transformation when $g = 1$.

3.3 Pyramidal Recurrent Unit

We extend the basic gating architecture of LSTM with the pyramidal and grouped linear transformations outlined above to produce the Pyramidal Recurrent Unit (PRU), whose improved sequence modeling capacity is evidenced in Section 4.

At time t , the PRU combines the input vector \mathbf{x}_t and the previous context vector (or previous hidden state vector) \mathbf{h}_{t-1} using the following transformation function as:

$$\hat{\mathcal{G}}_v(\mathbf{x}_t, \mathbf{h}_{t-1}) = \hat{\mathcal{F}}_P(\mathbf{x}_t) + \mathcal{F}_G(\mathbf{h}_{t-1}), \quad (5)$$

where $v \in \{f, i, c, o\}$ indexes the various gates in the LSTM model, and $\hat{\mathcal{F}}_P(\cdot)$ and $\mathcal{F}_G(\cdot)$ represents the pyramidal and grouped linear transformations defined in Eqns. 2 and 4, respectively.

We will now incorporate $\hat{\mathcal{G}}_v(\cdot, \cdot)$ into LSTM gating architecture to produce PRU. At time t , a PRU cell takes $\mathbf{x}_t \in \mathbb{R}^N$, $\mathbf{h}_{t-1} \in \mathbb{R}^M$, and $\mathbf{c}_{t-1} \in \mathbb{R}^M$ as inputs to produce forget \mathbf{f}_t , input \mathbf{i}_t , output \mathbf{o}_t , and content $\hat{\mathbf{c}}_t$ gate signals. The inputs are combined with these gate signals to produce context vector $\mathbf{h}_t \in \mathbb{R}^M$ and cell state $\mathbf{c}_t \in \mathbb{R}^M$. Mathematically, the PRU with the LSTM gating architecture can be defined as:

$$\begin{aligned} \mathbf{f}_t &= \sigma(\hat{\mathcal{G}}_f(\mathbf{x}_t, \mathbf{h}_{t-1})) \\ \mathbf{i}_t &= \sigma(\hat{\mathcal{G}}_i(\mathbf{x}_t, \mathbf{h}_{t-1})) \\ \hat{\mathbf{c}}_t &= \tanh(\hat{\mathcal{G}}_c(\mathbf{x}_t, \mathbf{h}_{t-1})) \\ \mathbf{o}_t &= \sigma(\hat{\mathcal{G}}_o(\mathbf{x}_t, \mathbf{h}_{t-1})) \\ \mathbf{c}_t &= \mathbf{f}_t \otimes \mathbf{c}_{t-1} + \mathbf{i}_t \otimes \hat{\mathbf{c}}_t \\ \mathbf{h}_t &= \mathbf{o}_t \otimes \tanh(\mathbf{c}_t) \end{aligned} \quad (6)$$

where \otimes represents the element-wise multiplication operation, and σ and \tanh are the sigmoid and hyperbolic tangent activation functions. We note that LSTM is a special case of PRU when $g=K=1$.

4 Experiments

To showcase the effectiveness of the PRU, we evaluate the performance on two standard datasets for word-level language modeling and compare with state-of-the-art methods. Additionally, we provide a detailed examination of the PRU and its behavior on the language modeling tasks.

4.1 Set-up

Dataset: Following recent works, we compare on two widely used datasets, the Penn Treebank (PTB) (Marcus et al., 1993) as prepared by Mikolov et al. (2010) and WikiText2 (WT-2) (Merity et al., 2017). For both datasets, we follow the same training, validation, and test splits as in Merity et al. (2018).

Language Model: We extend the language model, AWD-LSTM (Merity et al., 2018), by replacing LSTM layers with PRU. Our model uses 3-layers of PRU with an embedding size of 400. The number of parameters learned by state-of-the-art methods vary from 18M to 66M with majority of the methods learning about 22M to 24M parameters on the PTB dataset. For a fair comparison with state-of-the-art methods, we fix the model size to 19M and vary the value of g and hidden layer sizes so that total number of learned parameters is similar across different configurations. We use 1000, 1200, and 1400 as hidden layer sizes for values of $g=1,2$, and 4, respectively. We use the same settings for the WT-2 dataset. We set the number of pyramidal levels K to two in our experiments and use average pooling for sub-sampling. These values are selected based on our ablation experiments on the validation set (Section 4.4). We measure the performance of our models in terms of word-level perplexity. We follow the same training strategy as in Merity et al. (2018).

To understand the effect of regularization methods on the performance of PRUs, we perform experiments under two different settings: (1) *Standard dropout*: We use a standard dropout (Srivastava et al., 2014) with probability of 0.5 after embedding layer, the output between LSTM layers, and the output of final LSTM layer. (2) *Advanced dropout*: We use the same dropout techniques with the same dropout values as in Merity et al. (2018). We call this model as AWD-PRU.

4.2 Results

Table 1 compares the performance of the PRU with state-of-the-art methods. We can see that the PRU achieves the best performance with fewer parameters.

Standard dropout: PRUs achieve either the same or better performance than LSTMs. In particular, the performance of PRUs improves with the increasing value of g . At $g = 4$, PRUs out-

perform LSTMs by about 4 points on the PTB dataset and by about 3 points on the WT-2 dataset. This is explained in part by the regularization effect of the grouped linear transformation (Figure 1). With grouped linear and pyramidal transformations, PRUs learn rich representations at very high dimensional space while learning fewer parameters. On the other hand, LSTMs overfit to the training data at such high dimensions and learn $1.4\times$ to $1.8\times$ more parameters than PRUs.

Advanced dropouts: With the advanced dropouts, the performance of PRUs improves by about 4 points on the PTB dataset and 7 points on the WT-2 dataset. This further improves with finetuning on the PTB (about 2 points) and WT-2 (about 1 point) datasets.

Comparison with state-of-the-art: For similar number of parameters, the PRU with standard dropout outperforms most of the state-of-the-art methods by large margin on the PTB dataset (e.g. RAN (Lee et al., 2017) by 16 points with 4M less parameters, QRNN (Bradbury et al., 2017) by 16 points with 1M more parameters, and NAS (Zoph and Le, 2017) by 1.58 points with 6M less parameters). With advanced dropouts, the PRU delivers the best performance. On both datasets, the PRU improves the perplexity by about 1 point while learning 15-20% fewer parameters.

Inference: PRU is a drop-in replacement for LSTM, therefore, it can improve language models with modern inference techniques such as dynamic evaluation (Krause et al., 2018). When we evaluate PRU-based language models (only with standard dropout) with dynamic evaluation on the PTB test set, the perplexity of PRU ($g = 4, k = 2, M = 1400$) improves from 62.42 to 55.23 while the perplexity of an LSTM ($M = 1000$) with similar settings improves from 66.29 to 58.79; suggesting that modern inference techniques are equally applicable to PRU-based language models.

4.3 Analysis

It is shown above that the PRU can learn representations at higher dimensionality with more generalization power, resulting in performance gains for language modeling. A closer analysis of the impact of the PRU in a language modeling system reveals several factors that help explain how the PRU achieves these gains.

	WT-2			PTB		
Model	Params	Val	Test	Params	Val	Test
Variational LSTM (Gal and Ghahramani, 2016)	–	–	–	20 M	–	78.6
CharCNN (Kim et al., 2016)	–	–	–	19 M	–	78.9
Pointer Sentinel-LSTM (Merity et al., 2017)	–	–	–	19 M	72.4	70.9
RHN (Zilly et al., 2016)	–	–	–	23 M	67.9	65.4
NAS Cell (Zoph and Le, 2017)	–	–	–	25 M	–	64.0
Variational LSTM - (Inan et al., 2017)	28 M	91.5	87	24 M	75.7	73.2
SRU - 6 layers (Lei et al., 2018)	–	–	–	24 M	63.4	60.3
QRNN (Bradbury et al., 2017)	–	–	–	18 M	82.1	78.3
RAN (Lee et al., 2017)	–	–	–	22 M	–	78.5
4-layer skip-connection LSTM (Melis et al., 2018)	–	–	–	24 M	60.9	58.3
AWD-LSTM - (Merity et al., 2018)	33 M	69.1	66	24 M	60.7	58.8
AWD-LSTM - (Merity et al., 2018)-finetuned	33 M	68.6	65.8	24 M	60	57.3
Variational LSTM (Gal and Ghahramani, 2016)	–	–	–	66 M	–	73.4
NAS Cell (Zoph and Le, 2017)	–	–	–	54 M	–	62.4
Quantized LSTM - Full precision (Xu et al., 2018)	–	–	100.1	–	–	89.8
Quantized LSTM - 2 bit (Xu et al., 2018)	–	–	106.1	–	–	95.8
With standard dropout						
LSTM ($M = 1000$)	29 M	78.93	75.08	20 M	68.57	66.29
LSTM ($M = 1200$)	35 M	77.93	74.48	26 M	69.17	67.16
LSTM ($M = 1400$)	42 M	77.55	74.44	33 M	70.88	68.55
Ours -PRU ($g = 1, K = 2, M = 1000$)	28 M	79.15	76.59	19 M	69.8	67.78
Ours -PRU ($g = 2, K = 2, M = 1200$)	28 M	76.62	73.79	19 M	67.17	64.92
Ours -PRU ($g = 4, K = 2, M = 1400$)	28 M	75.46	72.77	19 M	64.76	62.42
With advanced dropouts						
Ours - AWD-PRU ($g = 1, K = 2, M = 1000$)	28 M	71.84	68.6	19 M	61.72	59.54
Ours - AWD-PRU ($g = 2, K = 2, M = 1200$)	28 M	68.57	65.7	19 M	60.81	58.65
Ours - AWD-PRU ($g = 4, K = 2, M = 1400$)	28 M	68.17	65.3	19 M	60.62	58.33
Ours - AWD-PRU ($g = 4, K = 2, M = 1400$)-finetuned	28 M	67.19	64.53	19 M	58.46	56.56

Table 1: Comparison of single model word-level perplexity of our model with state-of-the-art on validation and test sets of Penn Treebank and Wikitext-2 dataset. For evaluation, we select the model with minimum validation loss. Lower perplexity value represents better performance.

Confidence: As exemplified in Table 2a, the PRU tends toward more confident decisions, placing more of the probability mass on the top next-word prediction than the LSTM. To quantify this effect, we calculate the entropy of the next-token distribution for both the PRU and the LSTM using 3687 contexts from the PTB validation set. Figure 3 shows a histogram of the entropies of the distribution, where bins of size 0.23 are used to effect categories. We see that the PRU more often produces lower entropy distributions corresponding to higher confidences for next-token choices. This is evidenced by the mass of the red PRU curve lying in the lower entropy ranges compared to the blue LSTM’s curve. The PRU can produce confident decisions in part because more information is encoded in the higher dimensional context vectors.

Variance in word embeddings: The PRU has the ability to model individual words at different resolutions through the pyramidal transform; which provides multiple paths for the gradient to the embedding layer (similar to multi-task learning) and improves the flow of information. When considering the embeddings by part of speech, we find that the pyramid level 1 embeddings exhibit higher variance than the LSTM across all POS categories (Figure 4), and that pyramid level 2 embeddings show extremely low variance¹. We hypothesize that the LSTM must encode both coarse group similarities and individual word differences into the same vector space, reducing the space between individual words of the same category. The PRU can rely on the subsampled embeddings to

¹POS categories are computed using NLTK toolkit.

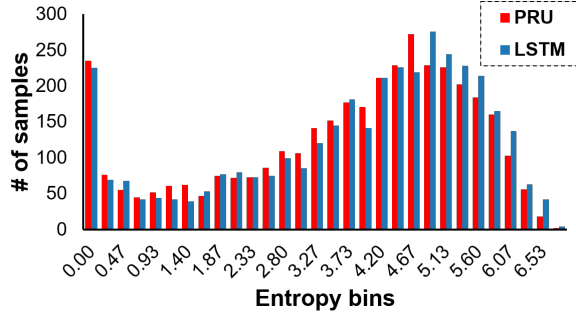


Figure 3: Histogram of the entropies of next-token distributions predicted by the PRU (mean 3.80) and the LSTM (mean 3.93) on the PTB validation set. Lower entropy values indicate higher confidence decisions, which is desirable if decisions are often correct.

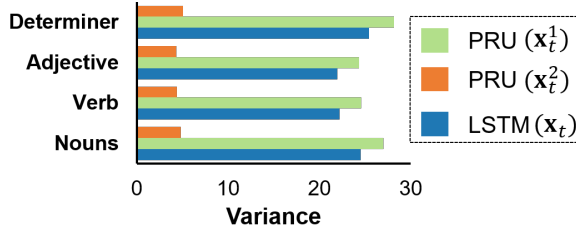


Figure 4: Variance of learned word embeddings for different categories of words on the PTB validation set. We compute the variance of a group of embeddings as the average squared euclidean distance to their mean. Higher variance may allow for better intra-category distinctions. The PRU with pyramid levels 1 and 2 is shown.

account for coarse-grained group similarities, allowing for finer individual word distinctions in the embedding layer. This hypothesis is strengthened by the entropy results described above: a model which can make finer distinctions between individual words can more confidently assign probability mass. A model that cannot make these distinctions, such as the LSTM, must spread its probability mass across a larger class of similar words.

Gradient-based analysis: Saliency analysis using gradients help identify relevant words in a test sequence that contribute to the prediction (Gevrey et al., 2003; Li et al., 2016; Arras et al., 2017). These approaches compute the relevance as the squared norm of the gradients obtained through back-propagation. Table 2a visualizes the heatmaps for different sequences. PRUs, in general, give more relevance to contextual words than

LSTMs, such as southeast (sample 1), cost (sample 2), face (sample 4), and introduced (sample 5), which help in making more confident decisions. Furthermore, when gradients during back-propagation are visualized (Selvaraju et al., 2017) (Table 2b), we find that PRUs have better gradient coverage than LSTMs, suggesting PRUs use more features than LSTMs that contributes to the decision. This also suggests that PRUs update more parameters at each iteration which results in faster training. Language model in (Merity et al., 2018) takes 500 and 750 epochs to converge with PRU and LSTM as a recurrent unit, respectively.

4.4 Ablation studies

In this section, we provide a systematic analysis of our design choices. Our training methodology is the same as described in Section 4.1 with the standard dropouts. For a thorough understanding of our design choices, we use a language model with a single layer of PRU and fix the size of embedding and hidden layers to 600. The word-level perplexities are reported on the validation sets of the PTB and the WT-2 datasets.

Pyramidal levels K and groups g : The two hyper-parameters that control the trade-off between performance and number of parameters in PRUs are the number of pyramidal levels K and groups g . Figure 5 provides a trade-off between perplexity and recurrent unit (RU) parameters².

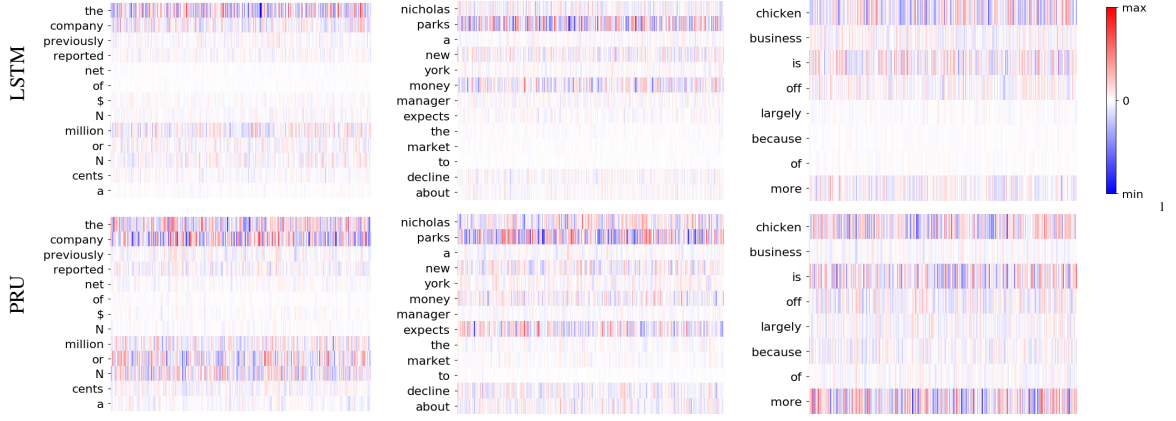
Variable K and fixed g : When we increase the number of pyramidal levels K at a fixed value of g , the performance of the PRU drops by about 1 to 4 points while reducing the total number of recurrent unit parameters by up to 15%. We note that the PRU with $K = 4$ at $g = 1$ delivers similar performance as the LSTM while learning about 15% fewer recurrent unit parameters.

Fixed K and variable g : When we vary the value of g at fixed number of pyramidal levels K , the total number of recurrent unit parameters decreases significantly with a minimal impact on the perplexity. For example, PRUs with $K = 2$ and $g = 4$ learns 77% fewer recurrent unit parameters while its perplexity (lower is better) increases by about 12% in comparison to LSTMs. Moreover, the decrease in number of parameters at higher value of g enables PRUs to learn the representations in high dimensional space with better generalizability (Table 1).

²# total params = # embedding params + # RU params

Gradient-based sensitivity analysis heatmaps		LSTM top-5	PRU top-5
	the tremor was centered near <unk> southeast of	asia 0.04 europe 0.04 america 0.04 south 0.06 the 0.31	asia 0.04 the 0.05 california 0.07 <unk> 0.11 san 0.12
LSTM			
PRU			
Reference: the tremor was centered near <unk> southeast of san francisco			
	the messages last N minutes and typically cost about	half 0.00 about 0.01 a 0.01 N 0.10 \$ 0.84	a 0.00 about 0.00 # 0.00 N 0.04 \$ 0.93
LSTM			
PRU			
Reference: the messages last N minutes and typically cost about \$ N.			
	he visits the same department every two or three	of 0.04 times 0.06 <eos> 0.06 years 0.09 <unk> 0.12	<unk> 0.03 times 0.04 months 0.10 weeks 0.10 years 0.38
LSTM			
PRU			
Reference: he visits the same department every two or three weeks .			
	but pipeline companies estimate they still face \$ N billion in liabilities from <unk> disputes including \$ N	<unk> 0.00 N 0.00 trillion 0.00 billion 0.38 million 0.62	<unk> 0.00 N 0.00 trillion 0.00 million 0.39 billion 0.60
LSTM			
PRU			
Reference: but pipeline companies estimate they still face \$ N billion in liabilities from <unk> disputes including \$ N billion .			
	chicken chains also are feeling more pressure from mcdonald's corp. which introduced its <unk> <unk> this	spring 0.05 summer 0.11 week 0.14 month 0.15 year 0.35	fall 0.04 week 0.08 summer 0.08 month 0.12 year 0.62
LSTM			
PRU			
Reference: chicken chains also are feeling more pressure from mcdonald's corp. which introduced its <unk> <unk> this year .			

(a) Gradient-based saliency analysis. Saliency score is proportional to cell coverage in red.



(b) Gradients during back-propagation for a test sequence (x-axis: dimensions of word vector, y-axis: test sequence)

Table 2: Qualitative comparison between the LSTM and the PRU: (a) Gradient-based saliency analysis along with top-5 predicted words. (b) Gradients during back-propagation. For computing the gradients for a given test sequence, the top-1 predicted word was used as the *true* predicted word. Best viewed in color.

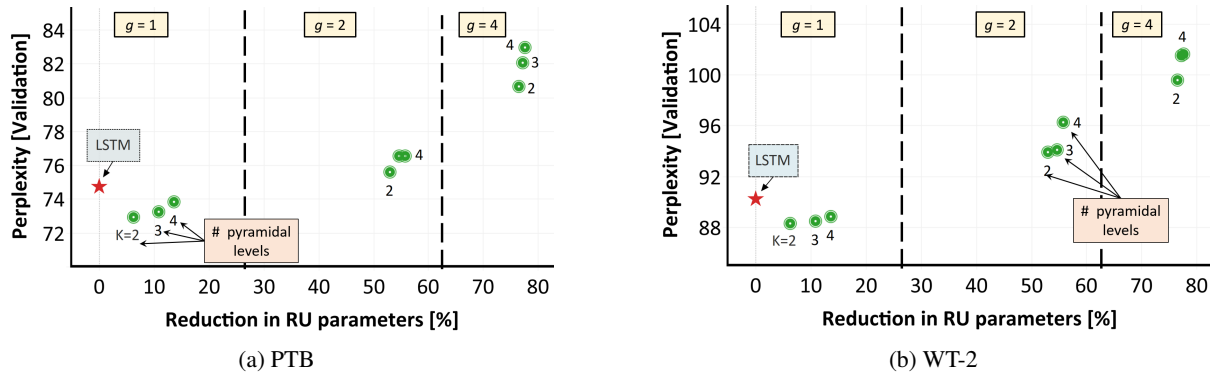


Figure 5: Impact of number of groups g and pyramidal levels K on the perplexity. Reduction in recurrent unit (RU) parameters is computed with respect to LSTM. Lower perplexity value represents better performance.

Transformations: Table 3 shows the impact of different transformations of the input vector \mathbf{x}_t and the context vector \mathbf{h}_{t-1} . We make following observations: (1) Using the pyramidal transformation for the input vectors improves the perplexity by about 1 point on both the PTB and WT-2 datasets while reducing the number of recurrent unit parameters by about 14% (see R1 and R4). We note that the performance of the PRU drops by up to 1 point when residual connections are not used (R4 and R6). (2) Using the grouped linear transformation for context vectors reduces the total number of recurrent unit parameters by about 75% while the performance drops by about 11% (see R3 and R4). When we use the pyramidal transformation instead of the linear transformation, the performance drops by up to 2% while there is no significant drop in the number of parameters (R4 and R5).

Subsampling: We set sub-sampling kernel κ (Eq. 3) with stride $s = 2$ and size of 3 ($e = 1$) in four different ways: (1) *Skip*: We skip every other element in the input vector. (2) *Convolution*: We initialize the elements of κ randomly from normal distribution and learn them during training the model. We limit the output values between -1 and 1 using *tanh* activation function to make training stable. (3) *Avg. pool*: We initialize the elements of κ to $\frac{1}{3}$. (4) *Max pool*: We select the maximum value in the kernel window κ .

Table 4 compares the performance of the PRU with different sampling methods. Average pooling performs the best while skipping give comparable

	Transformations		PTB		WT-2	
			PPL	# Params (total/RU)	PPL	# Params (total/RU)
	Context	Input				
R1	LT	LT	74.80	8.8/2.9	89.30	22.8/2.9
R2	GLT	GLT	84.38	6.5/0.5	104.13	20.46/0.5
R3	GLT	PT	82.67	6.6/0.64	99.57	20.6/0.64
R4	LT	PT	74.18	8.5/2.5	88.31	22.5/2.5
R5	PT	PT	75.80	8.1/2.1	90.56	22.1/2.1
R6	LT	PT [†]	75.61	8.5/2.5	89.27	22.5/2.5

Table 3: Impact of different transformations used for processing input and context vectors (LT - linear transformation, PT - pyramidal transformation, and GLT - grouped linear transformation). Here, [†] represents that PT was used without residual connection, PPL represents word-level perplexity (lower is better), and the number of parameters are in million. We used $K=g=4$ in our experiments.

Dataset	Skip	Max pool	Avg. Pool	Convolution
PTB	75.12	87.6	73.86	81.56
WT-2	89.24	107.63	88.88	93.16

Table 4: Impact of different sub-sampling methods on the word-level perplexity (lower is better). We used $g=1$ and $K=4$ in our experiments.

performance. Both of these methods enable the network to learn richer word representations while representing the input vector in different forms, thus delivering higher performance. Surprisingly, a convolution-based sub-sampling method does not perform as well as the averaging method. The *tanh* function used after convolution limits the range of output values which are further limited by the LSTM gating structure, thereby impeding in the flow of information inside the cell. Max pooling forces the network to learn representations from high magnitude elements, thus distinguishing features between elements vanishes, resulting in poor performance.

5 Conclusion

We introduce the Pyramidal Recurrent Unit, which better model contextual information by admitting higher dimensional representations with good generalizability. When applied to the task of language modeling, PRUs improve perplexity across several settings, including recent state-of-the-art systems. Our analysis shows that the PRU improves the flow of gradient and expand the word embedding subspace, resulting in more confident decisions. Here we have shown improvements for language modeling. In future, we plan to study the performance of PRUs on different tasks, including machine translation and question answering. In addition, we will study the performance of the PRU on language modeling with more recent inference techniques, such as dynamic evaluation and mixture of softmax.

Acknowledgments

This research was supported by NSF (IIS 1616112, III 1703166), Allen Distinguished Investigator Award, and gifts from Allen Institute for AI, Google, Amazon, and Bloomberg. We are grateful to Aaron Jaech, Hannah Rashkin, Mandar Joshi, Aniruddha Kembhavi, and anonymous reviewers for their helpful comments.

References

- Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017. Explaining recurrent neural network predictions in sentiment analysis. In *8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*.
- James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2017. Quasi-recurrent neural networks. In *International Conference for Learning Representations (ICLR)*.
- Peter J Burt and Edward H Adelson. 1987. The laplacian pyramid as a compact image code. In *Readings in Computer Vision*. Elsevier.
- Víctor Campos, Brendan Jou, Xavier Giró-i Nieto, Jordi Torres, and Shih-Fu Chang. 2018. Skip rnn: Learning to skip state updates in recurrent neural networks. In *International Conference for Learning Representations (ICLR)*.
- Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.
- Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems (NIPS)*.
- Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International Conference on Machine Learning (ICML)*.
- Felix A Gers and Jürgen Schmidhuber. 2000. Recurrent nets that time and count. In *IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN)*.
- Muriel Gevrey, Ioannis Dimopoulos, and Sovan Lek. 2003. Review and comparison of methods to study the contribution of variables in artificial neural network models. *Ecological modelling*.
- Dongyoon Han, Jiwhan Kim, and Junmo Kim. 2017. Deep pyramidal residual networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition (CVPR)*.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.
- Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *International Conference for Learning Representations (ICLR)*.
- Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Association for the Advancement of Artificial Intelligence (AAAI)*.
- Diederik P Kingma, Tim Salimans, and Max Welling. 2015. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems (NIPS)*.
- Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. 2018. Dynamic evaluation of neural sequence models. In *International Conference on Machine Learning (ICML)*.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NIPS)*.
- Oleksii Kuchaiev and Boris Ginsburg. 2017. Factorization tricks for lstm networks. In *International Conference for Learning Representations (ICLR) Workshop*.
- Yann LeCun, Yoshua Bengio, et al. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*.
- Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2017. Recurrent additive networks. *arXiv preprint arXiv:1705.07393*.
- Tao Lei, Yu Zhang, and Yoav Artzi. 2018. Training rnns as fast as cnns. In *Empirical Methods in Natural Language Processing (EMNLP)*.
- Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.
- David G Lowe. 1999. Object recognition from local scale-invariant features. In *IEEE international conference on Computer vision (ICCV)*.
- Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*.
- Sachin Mehta, Mohammad Rastegari, Anat Caspi, Linda Shapiro, and Hannaneh Hajishirzi. 2018. Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In *European Conference in Computer Vision (ECCV)*.
- Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. In *International Conference for Learning Representations (ICLR)*.
- Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing lstm language models. In *International Conference for Learning Representations (ICLR)*.

-
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference for Learning Representations (ICLR)*.
- Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.
- Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE International Conference on Computer Vision (ICCV)*.
- Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Query-reduction networks for question answering. In *International Conference on Learning Representations (ICLR)*.
- Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2018. Neural speed reading via skim-rnn. In *International Conference on Learning Representations (ICLR)*.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research (JMLR)*.
- Chen Xu, Jianqiang Yao, Zhouchen Lin, Wenwu Ou, Yuanbin Cao, Zhirong Wang, and Hongbin Zha. 2018. Alternating multi-bit quantization for recurrent neural networks. In *International Conference for Learning Representations (ICLR)*.
- Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W Cohen. 2018. Breaking the softmax bottleneck: a high-rank rnn language model. In *International Conference for Learning Representations (ICLR)*.
- Adams Wei Yu, Hongrae Lee, and Quoc Le. 2017. Learning to skim text. In *Association for Computational Linguistics (ACL)*.
- Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. 2016. Recurrent highway networks. *arXiv preprint arXiv:1607.03474*.
- Barret Zoph and Quoc V Le. 2017. Neural architecture search with reinforcement learning. In *International Conference for Learning Representations (ICLR)*.