Due Date: March 17th 23:00, 2020

Instructions

- *For all questions, show your work!*
- *Submit your report (pdf) and your code electronically via the course Gradescope page.*
- *An outline of code will be provided in the course repo at this link. You must start from this outline and follow the instructions in it (even if you use different code, you must follow the overall outline and instructions).*
- *TAs for this assignment are Jessica Thompson, Jonathan Cornford and Lluis Castrejon.*

Summary:

In this assignment, you will implement and train **sequential language models** on the Penn Treebank dataset. Language models learn to assign a likelihood to sequences of text. The elements of the sequence (typically words or individual characters) are called tokens, and can be represented as one-hot vectors with length equal to the vocabulary size, e.g. 26 for a vocabulary of English letters with no punctuation or spaces, in the case of characters, or as indices in the vocabulary for words. In this representation an entire dataset (or a mini-batch of examples) can be represented by a 3-dimensional tensor, with axes corresponding to: (1) the example within the dataset/mini-batch, (2) the time-step within the sequence, and (3) the index of the token in the vocabulary. Sequential language models do **next-step prediction**, in other words, they predict tokens in a sequence one at a time, with each prediction based on all the previous elements of the sequence. A trained sequential language model can also be used to generate new sequences of text, by making each prediction conditioned on the past *predictions* (instead of the ground-truth input sequence).

As a starting point, you are provided with an implementation of a **simple ("vanilla") RNN** (recurrent neural network). Problem 1 is to implement an RNN with a gating mechanism on the hidden state, specifically with **gated recurrent units (GRUs)**. Problem 2 is to implement the **attention module of a transformer network** (we provide you with PyTorch code for the rest of the transformer). Problem 3 is to train these 3 models using a variety of different optimizers and hyperparameter settings and Problem 4 is to analyze the behaviour of the trained models. Each problem is worth 25 points.

The Penn Treebank Dataset This is a dataset of about 1 million words from about 2,500 stories from the Wall Street Journal. It has Part-of-Speech annotations and is sometimes used for training parsers, but it's also a very common benchmark dataset for training RNNs and other sequence models to do next-step prediction.

Preprocessing: The version of the dataset you will work with has been preprocessed: lower-cased, stripped of non-alphabetic characters, tokenized (broken up into words, with sentences separated by the `<eos>` (end of sequence) token), and cut down to a vocabulary of 10,000 words; any word not in this vocabulary is replaced by `<unk>`. For the transformer network, positional information (an embedding of the position in the source sequence) for each token is also included in the input sequence. In both cases the preprocessing code is given to you.

Problem 1

Implementing an RNN with Gated Recurrent Units (GRU) (25pts) The implementation of your RNN must be able to process mini-batches. Implement the model **from scratch** using PyTorch Tensors, Variables, and associated operations (e.g. as found in the `torch.nn` module). Specifically, use appropriate matrix and tensor operations (e.g. dot, multiply, add, etc.) to implement the recurrent unit calculations; you **may not** use built-in Recurrent modules. You **may** subclass `nn.module`, use built-in Linear modules, and built-in implementations of nonlinearities (tanh, sigmoid, and softmax), initializations, loss functions, and optimization algorithms. Your code must start from the code scaffold and follow its structure and instructions.

The use of "gating" (i.e. element-wise multiplication, represented by the $\odot$ symbol) can significantly improve the performance of RNNs. The Long-Short Term Memory (LSTM) RNN is the best known example of gating in RNNs; GRU-RNNs are a slightly simpler variant (with fewer gates).

The equations for a GRU are:

$$\boldsymbol{r}_t = \sigma_r(\boldsymbol{W}_r\boldsymbol{x}_t + \boldsymbol{U}_r\boldsymbol{h}_{t-1} + \boldsymbol{b}_r) \tag{1}$$
$$\boldsymbol{z}_t = \sigma_z(\boldsymbol{W}_z\boldsymbol{x}_t + \boldsymbol{U}_z\boldsymbol{h}_{t-1} + \boldsymbol{b}_z) \tag{2}$$
$$\tilde{\boldsymbol{h}}_t = \sigma_h(\boldsymbol{W}_h\boldsymbol{x}_t + \boldsymbol{U}_h(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}) + \boldsymbol{b}_h) \tag{3}$$
$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t \tag{4}$$
$$P(\boldsymbol{y}_t|\boldsymbol{x}_1, ..., \boldsymbol{x}_t) = \sigma_y(\boldsymbol{W}_y\boldsymbol{h}_t + \boldsymbol{b}_y) \tag{5}$$

$\boldsymbol{r}_t$ is called the "reset gate" and $\boldsymbol{z}_t$ the "forget gate". The trainable parameters are $\boldsymbol{W}_r, \boldsymbol{W}_z, \boldsymbol{W}_h, \boldsymbol{W}_y$, $\boldsymbol{U}_r, \boldsymbol{U}_z, \boldsymbol{U}_h, \boldsymbol{b}_r, \boldsymbol{b}_z, \boldsymbol{b}_h$, and $\boldsymbol{b}_y$, as well as the initial hidden state parameter $\boldsymbol{h}_0$. GRUs use the sigmoid activation function for σ_r and σ_z, and tanh for σ_h.

See further instructions in the solution template.

Solution
Submitted to gradescope

Problem 2

Implementing the attention module of a transformer network (25pts) While prototypical RNNs "remember" past information by taking their previous hidden state as input at each step, recent years have seen a profusion of methodologies for making use of past information in different ways. The transformer [1] is one such fairly new architecture which uses several self-attention networks

[1] See `https://arxiv.org/abs/1706.03762` for more details.

("heads") in parallel, among other architectural specifics. The transformer is quite complicated to implement compared to the RNNs described so far; most of the code is provided and your task is only to implement the multi-head scaled dot-product attention. The attention vector for m heads indexed by i is calculated as follows:

$$\boldsymbol{A}_i = \text{softmax}\left(\frac{\boldsymbol{Q}_i\boldsymbol{W}_{Q_i}(\boldsymbol{K}_i\boldsymbol{W}_{K_i})^\top}{\sqrt{d_k}}\right) \tag{6}$$

$$\boldsymbol{H}_i = \boldsymbol{A}_i\boldsymbol{V}\boldsymbol{W}_{V_i} \tag{7}$$

$$A(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = \text{concat}(\boldsymbol{H}_1,...,\boldsymbol{H}_m)\boldsymbol{W}_O \tag{8}$$

where $\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}$ are queries, keys, and values respectively, $\boldsymbol{W}_{Q_i},\boldsymbol{W}_{K_i},\boldsymbol{W}_{V_i}$ are their corresponding embedding matrices, $\boldsymbol{W}_O$ is the output embedding, and d_k is the dimension of the keys. $\boldsymbol{Q},\boldsymbol{K}$, and $\boldsymbol{V}$ are determined by the output of the feed-forward layer of the main network (given to you). $\boldsymbol{A}_i$ are the attention values, which specify which elements of the input sequence each attention head attends to.

Note that the implementation of multi-head attention requires binary masks, so that attention is computed only over the past, not the future. A mask value of 1 indicates an element which the model is allowed to attend to (i.e. from the past); a value of 0 indicates an element it is not allowed to attend to. This can be implemented by modifying the softmax function to account for the mask $\boldsymbol{s}$ as follows:

$$\tilde{\boldsymbol{x}} = \exp(\boldsymbol{x}) \odot \boldsymbol{s} \tag{9}$$

$$\text{softmax}(\boldsymbol{x},\boldsymbol{s}) \doteq \frac{\tilde{\boldsymbol{x}}}{\sum_i \tilde{x}_i} \tag{10}$$

To avoid potential numerical stability issues, we recommend a different implementation:

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} \odot \boldsymbol{s} - 10^9(1-\boldsymbol{s}) \tag{11}$$

$$\text{softmax}(\boldsymbol{x},\boldsymbol{s}) \doteq \frac{\exp(\tilde{\boldsymbol{x}})}{\sum_i \exp(\tilde{x}_i)} \tag{12}$$

This second version is equivalent (up to numerical precision) as long as $\boldsymbol{x} >> -10^9$, which should be the case in practice.

Solution
Submitted to gradescope

Problem 3

Training language models and model comparison (25pts) Unlike in classification problems, where the performance metric is typically accuracy, in language modelling, the performance metric is typically based directly on the cross-entropy loss, i.e. the negative log-likelihood (NLL) the model assigns to the tokens. For word-level language modelling it is standard to report **perplexity (PPL)**, which is the exponentiated average per-token NLL (over all tokens):

$$\exp\left(\frac{1}{TN}\sum_{t=1}^{T}\sum_{n=1}^{N} -\log P(\boldsymbol{x}_t^{(n)}|\boldsymbol{x}_1^{(n)},....,\boldsymbol{x}_{t-1}^{(n)})\right),$$

where t is the index with the sequence, and n indexes different sequences. For Penn Treebank in particular, the test set is treated as a single sequence (i.e. $N = 1$). The purpose of this assignment is to perform model exploration, which is done using a validation set. As such, we do not require you to run your models on the test set.

You will train each of the three architectures using either stochastic gradient descent or the ADAM optimizer. The training loop is provided in *run_exp.py*.

1. - 4. You are asked to run 4 experiments (3.1, 3.2, 3.3, 3.4) with different architectures, optimizers, and hyperparameters settings. These parameter settings are given to you in the code (*run_exp.py*). In total there are 15 settings for you to run ($5 + 3 + 3 + 4 = 15$). For each experiment (3.1, 3.2, 3.3, 3.4), plot learning curves (train and validation) of PPL over both **epochs** and **wall-clock-time**. Figures should have labeled axes and a legend and an explanatory caption.
Solution

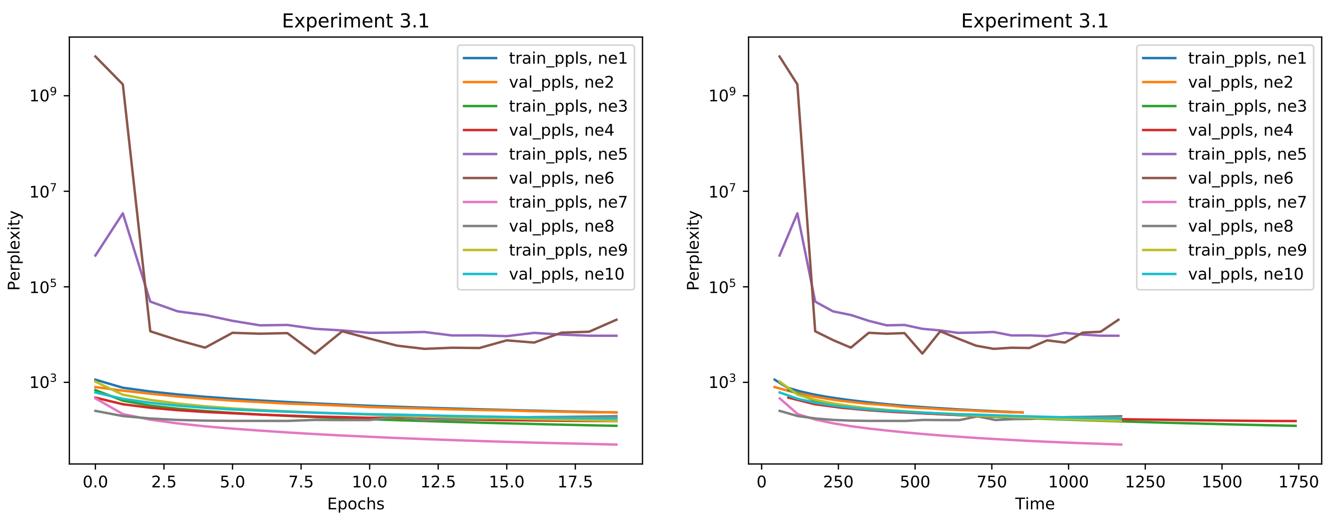

Figure 1: Experiment 3.1. Train and validation perplexity of 5 settings (10 experiments because each setting corresponds for 1 experiment of training and 1 for validation) vs epochs (left figure) and time (right figure)

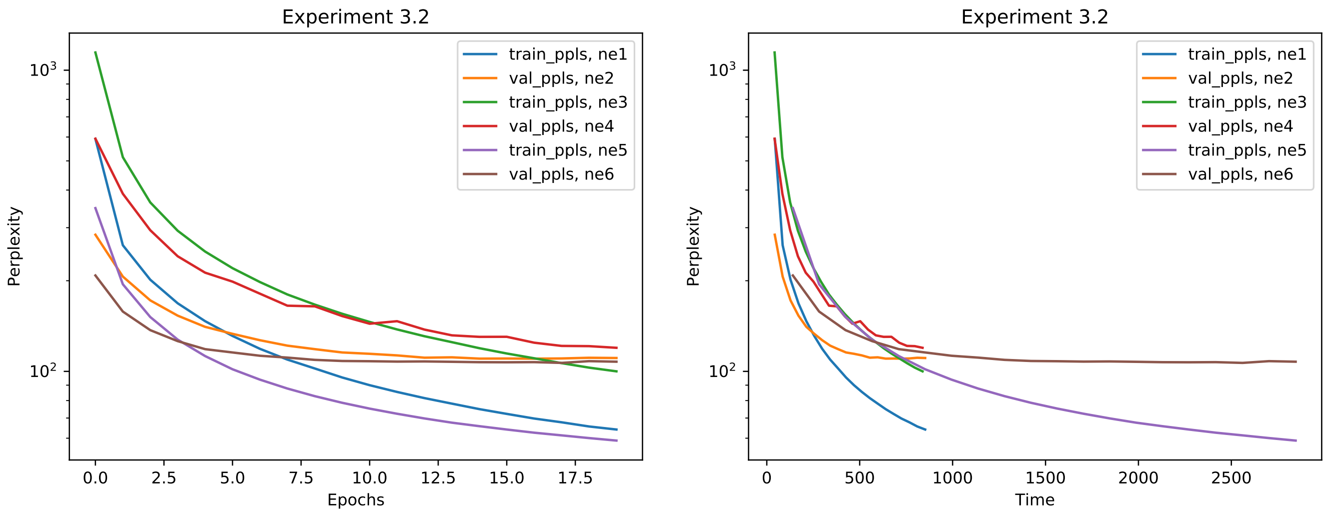

Figure 2: Experiment 3.2. Train and validation perplexity of 3 settings (6 experiments because each setting corresponds for 1 experiment of training and 1 for validation) vs epochs (left figure) and time (right figure)

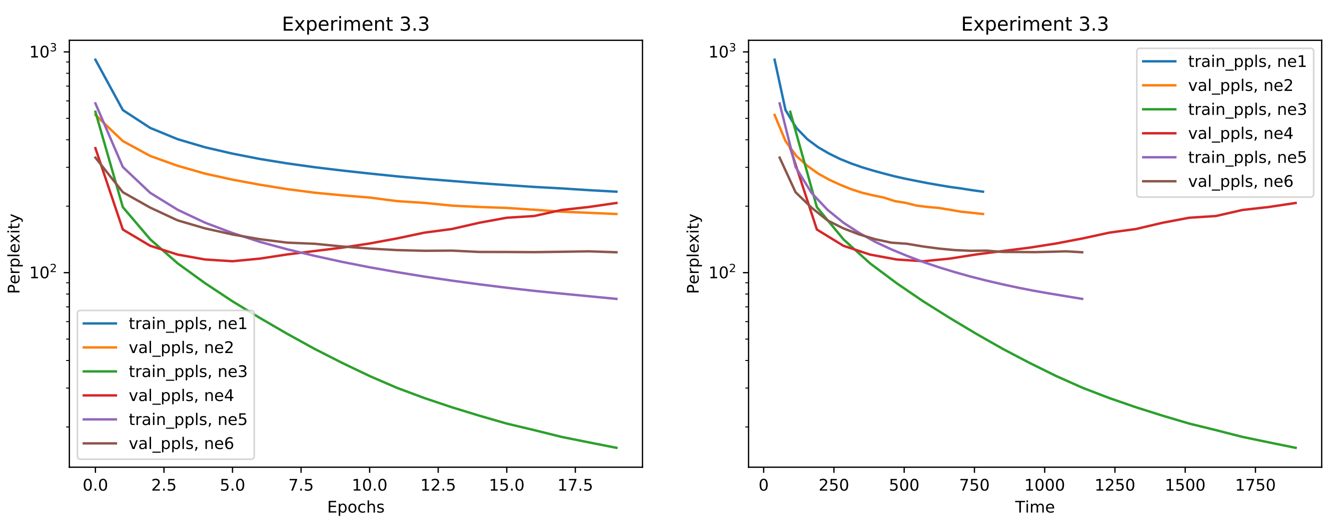

Figure 3: Experiment 3.3. Train and validation perplexity of 3 settings (6 experiments because each setting corresponds for 1 experiment of training and 1 for validation) vs epochs (left figure) and time (right figure)

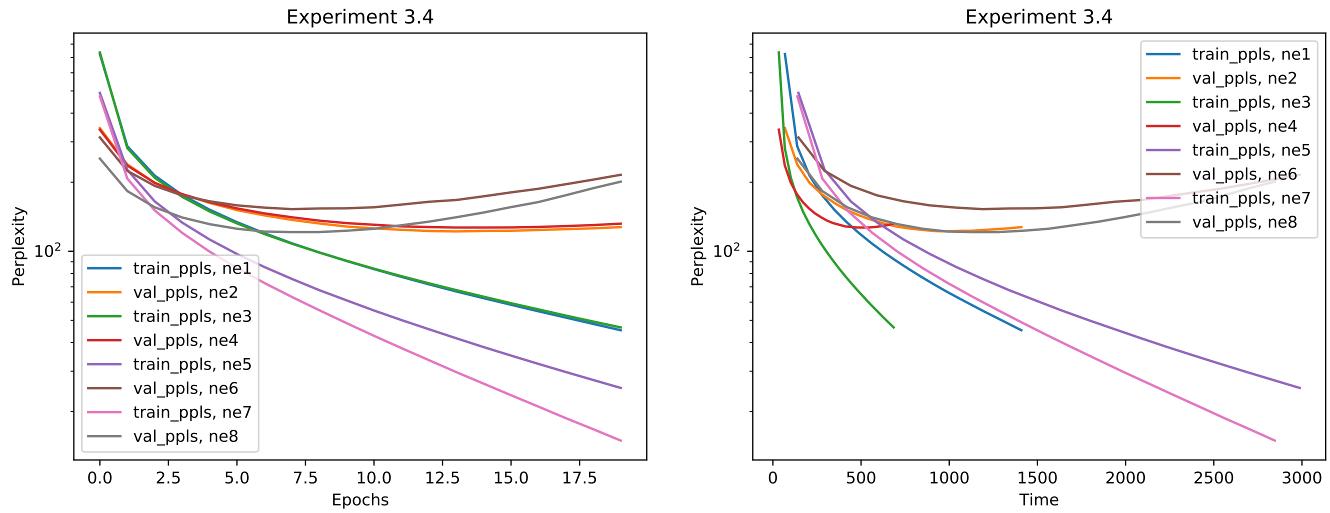

Figure 4: Experiment 3.4. Train and validation perplexity of 4 settings (8 experiments because each setting corresponds for 1 experiment of training and 1 for validation)vs epochs (left figure) and time (right figure)

5. Make a table of results summarizing the train and validation performance for each experiment, indicating the architecture and optimizer. Sort by architecture, then optimizer, and number the experiments to refer to them easily later. Bold the best result for each architecture.[2] The table should have an explanatory caption, and appropriate column and/or row headers. Any shorthand or symbols in the table should be explained in the caption.

   Solution

   The notation using the tables is: **ne:** Number of experiment. **arch:** Architecture. **opt:** optimizer. **lr:** learning rate. **bs:** batch size. **hs:** hidden size. **nl:** number of layers. **dkp:** dropout keep probability. **time:** final time of the experiment. **ep:** epochs.

ne	arch	opt	lr	bs	hs	nl	dkp	train/val	time	ep1	ep2	ep3	ep4	ep5
1	RNN	SGD	1.0	128	512	2	0.8	train	850.08	1139.62	771.79	644.65	559.61	498.97
2	RNN	SGD	1.0	128	512	2	0.8	valid	850.08	796.77	670.16	578.56	505.48	454.2
3	RNN	SGD	1.0	20	512	2	0.8	train	1739.86	690.08	413.47	326.23	281.03	251.1
4	RNN	SGD	1.0	20	512	2	0.8	valid	1739.86	479.05	348.39	294.16	261.42	239.69
5	RNN	SGD	10	128	512	2	0.8	train	1162.45	451490	3460379.4	49035.55	30640.19	25753.03
6	RNN	SGD	10	128	512	2	0.8	valid	1162.45	66380865	172724441	11733.56	7680.19	5327.88
7	RNN	ADAM	0.001	128	512	2	0.8	train	1171.39	462.59	216.8	165.7	138.18	120.19
8	RNN	ADAM	0.001	128	512	2	0.8	valid	1171.39	252.93	197	175.26	163.35	157.89
9	RNN	ADAM	0.0001	128	512	2	0.8	train	1170.11	1035.09	550.61	429.04	360.57	316.22
10	RNN	ADAM	0.0001	128	512	2	0.8	valid	1170.11	614	452.92	373.73	325.7	293.64

Figure 5: Train and validation performance experiment 3.1

[2]You can also make the table in LaTeX, but you can also make it using Excel, Google Sheets, or a similar program, and include it as an image.

ne	ep6	ep7	ep8	ep9	ep10	ep11	ep12	ep13	ep14	ep15	ep16	ep17	ep18	ep19	ep20
1	454.03	417.63	387.96	362.78	341.33	323.56	308.42	295.25	283.78	273.4	264.12	255.41	247.69	240.78	234.01
2	413.62	386.22	359.02	343.06	325.15	304.26	292.37	283.79	271.69	263.76	257.95	248.4	242.81	236.2	235.47
3	229.34	212.29	198.48	187.41	177.53	169.1	161.56	155.1	148.92	143.58	138.76	134.34	130.17	126.35	122.94
4	225.11	211.46	201.77	192.86	187.2	180.87	182.9	171.15	167.43	164.97	161.65	159.41	157.17	155.7	154.89
5	19411.74	15626.92	15910.66	13209.28	12230.33	10861.42	11029.66	11337.93	9593.62	9630.32	9284.27	10859.74	9996.92	9473.33	9457.55
6	10896.56	10432.94	10742.85	3993.63	11729.53	8244.19	5871.26	5026.05	5307.15	5234.92	7574.3	6807.02	10998.85	11457.78	20443.02
7	107.42	97.71	89.49	82.69	77.12	72.6	68.38	64.94	61.92	59.16	56.85	54.76	53.04	51.46	49.85
8	156.09	156.12	155.73	163.53	162.33	162.41	194.11	162.62	169.95	171.62	183.07	187.46	189.08	192.32	196.12
9	285.46	263.94	247.57	233.61	221.59	211.38	202.4	194.7	187.51	181.29	175.62	169.95	164.86	160.47	156.18
10	270.77	255.54	242.32	231.65	223.64	215.46	210.93	204.43	198.34	193.52	189.23	185.39	181.13	178.34	175.55

Figure 6: Train and validation performance experiment 3.1

ne	arch	opt	lr	bs	hs	nl	dkp	train/val	time	ep1	ep2	ep3	ep4	ep5
1	GRU	ADAM	0.001	128	512	2	0.5	train	852.37	591.52	262.38	201.57	168.33	146.82
2	GRU	ADAM	0.001	128	512	2	0.5	valid	852.37	284.31	206.47	172.12	153.15	140.58
3	GRU	SGD	10	128	512	2	0.5	train	838.25	1145.12	514.32	363.94	293.07	250
4	GRU	SGD	10	128	512	2	0.5	valid	838.25	592.46	388.91	294.28	241.13	212.71
5	GRU	ADAM	0.001	20	512	2	0.5	train	2846	348.42	194.68	151.54	127.5	112.44
6	GRU	ADAM	0.001	20	512	2	0.5	valid	2846	208.22	158	137.08	126.03	118.54

Figure 7: Train and validation performance experiment 3.2

ne	ep6	ep7	ep8	ep9	ep10	ep11	ep12	ep13	ep14	ep15	ep16	ep17	ep18	ep19	ep20
1	131.13	118.82	109.44	102.25	95.44	89.99	85.46	81.52	78.15	74.95	72.26	69.74	67.76	65.64	64.12
2	133.27	126.9	121.75	118.54	115.57	114.4	113	111.05	111.36	110.26	110.34	110.22	110.4	110.98	110.77
3	220.13	197.97	180.05	166.64	155.35	145.95	137.82	130.84	125.03	119.41	114.66	110.54	106.52	102.92	100.04
4	198.63	181.14	165.25	164.43	152.63	143.97	146.79	137.65	131.74	130.22	130.29	124.54	121.51	121.29	119.76
5	101.63	93.89	87.72	82.75	78.66	75.24	72.34	69.78	67.57	65.78	64.12	62.59	61.32	60.06	58.92
6	115.57	112.7	111.12	109.21	108.28	108.08	107.77	107.91	107.62	107.29	107.21	107.33	106.71	108.15	107.69

Figure 8: Train and validation performance experiment 3.2

ne	arch	opt	lr	bs	hs	nl	dkp	train/val	time	ep1	ep2	ep3	ep4	ep5
1	GRU	ADAM	0.001	128	256	2	0.2	train	780.48	921.85	545.16	452.45	402.32	369.78
2	GRU	ADAM	0.001	128	256	2	0.2	valid	780.48	518.43	394.54	337.73	304.86	280.9
3	GRU	ADAM	0.001	128	2048	2	0.5	train	1892.98	535.43	198.04	141.53	110.19	89.57
4	GRU	ADAM	0.001	128	2048	2	0.5	valid	1892.98	366.34	156.82	132.31	120.75	114.73
5	GRU	ADAM	0.001	128	512	4	0.5	train	1133.39	584.33	301.09	229.84	192.7	168.4
6	GRU	ADAM	0.001	128	512	4	0.5	valid	1133.39	331.97	231.43	197.04	172.51	158.69

Figure 9: Train and validation performance experiment 3.3

ne	ep6	ep7	ep8	ep9	ep10	ep11	ep12	ep13	ep14	ep15	ep16	ep17	ep18	ep19	ep20
1	346.1	327.31	312.54	300.31	290.14	281.22	273.05	266.15	260.19	254.39	249.3	244.56	240.79	236.41	232.69
2	263.77	250.25	238.94	230.24	224.06	218.87	210.9	207.05	201.23	198.38	196.43	192.76	188.91	186.68	184.38
3	74.14	62.32	52.8	45.1	39	34.01	30.07	27.03	24.56	22.49	20.72	19.35	18.03	17.02	16.09
4	112.65	115.78	120.94	125.34	129.78	135.57	142.92	151.81	157.64	168.3	177.34	180.52	192.27	198.41	206.86
5	151.2	137.82	127.55	119.12	111.87	105.77	100.55	95.96	92.02	88.53	85.44	82.71	80.37	78.19	76.06
6	148.91	141.74	136.74	135.14	131.41	128.56	126.54	125.61	125.85	124.11	124.06	123.85	124.32	124.91	123.69

Figure 10: Train and validation performance experiment 3.3

ne	arch	opt	lr	bs	hs	nl	dkp	train/val	time	ep1	ep2	ep3	ep4	ep5
1	TRANSFOR	ADAM	0.0001	128	512	6	0.9	train	1408.86	724.06	287.23	213.76	175.97	151.86
2	TRANSFOR	ADAM	0.0001	128	512	6	0.9	valid	1408.86	344.17	239.29	198.79	177.15	162.57
3	TRANSFOR	ADAM	0.0001	128	512	2	0.9	train	684.84	735.64	281.68	209.22	172.74	149.22
4	TRANSFOR	ADAM	0.0001	128	512	2	0.9	valid	684.84	338.95	236.52	198.75	177.52	163.59
5	TRANSFOR	ADAM	0.0001	128	2048	2	0.6	train	2986.9	490.29	226.6	165	133.44	112.57
6	TRANSFOR	ADAM	0.0001	128	2048	2	0.6	valid	2986.9	313.58	224.16	192.88	175.21	165.15
7	TRANSFOR	ADAM	0.0001	128	1024	6	0.9	valid	2845.88	473.27	207.57	150.71	120.38	99.7
8	TRANSFOR	ADAM	0.0001	128	1024	6	0.9	valid	2845.88	253.55	183.03	155.51	140.39	131.3

Figure 11: Train and validation performance experiment 3.4

ne	ep6	ep7	ep8	ep9	ep10	ep11	ep12	ep13	ep14	ep15	ep16	ep17	ep18	ep19	ep20
1	133.93	119.91	108.4	98.83	90.7	83.61	77.35	71.79	66.82	62.45	58.46	54.74	51.39	48.22	45.27
2	151.19	143.4	137.06	132.23	128.19	125.8	124.09	122.58	122.07	122.65	122.85	123.84	124.87	125.9	127.48
3	132.3	118.94	107.94	98.82	90.91	83.95	77.89	72.51	67.7	63.37	59.36	55.81	52.42	49.38	46.57
4	153.79	146.28	140.69	135.99	132.67	130.51	128.37	127.54	126.93	126.92	127.04	127.66	128.75	130.08	131.94
5	97.34	85.39	75.8	67.71	61.1	55.18	50.14	45.74	41.76	38.22	35.1	32.28	29.75	27.47	25.37
6	158.54	155	152.64	153.67	153.92	155.53	159.77	164.35	167.32	173.64	180.6	187.29	196.05	205.3	215.48
7	84.55	72.75	63.28	55.37	48.61	42.8	37.84	33.54	29.74	26.48	23.57	21.03	18.72	16.76	14.95
8	125.38	121.88	121.23	121.26	122.9	125.31	129.84	134.61	140.68	147.56	155.8	163.91	175.52	188.41	201.11

Figure 12: Train and validation performance experiment 3.4

6. Which hyperparameters + optimizer would you use if you were most concerned with wall-clock time, with generalization performance.
Solution
Looking on the column of maximal time on Fig.5, Fig.7, Fig.9 and Fig.11, one realised that 2 models with low wall-clock time and good generalization performance are:

   - $ne = 3$ of experiment 3.4. i.e: TRANSFORMER with ADAM optimizer, $l_r = 0.0001$ batch size 128, 512 of hidden size, 2 layers, and $dkp = 0.9$
   - $ne = 1$ of experiment 3.3. i.e: GRU with ADAM optimizer, $l_r = 0.001$ batch size 128, 256 of hidden size, 2 layers, and $dkp = 0.2$

   In general terms increasing the mini-batch size could reduce the wall-clock training time, however it must not be very big, because we could degrade the quality of the model, measured by its ability to generalize. On the other hand, if we choose SGD we must use a not very low value for l_r. On the other hand, the wall-clock training time with ADAM is not strongly affected by variations in the learning rate because it is an adaptive method. If we are concern with the computational time, we must avoid architectures with a big quantity of layers and a large hidden size.

7. For exp 3.1 you trained an RNN with either SGD or ADAM. What did you notice about the optimizer's performance with different learning rates?
Solution

   We observe that the convergence is less influenced by the learning value (l_r) when one use ADAM optimizer. SGD on the other hand is strongly affected by l_r, for example experiments $1, 2$ and $5, 6$ in Fig.5 only differ in the learning rate value. For $1, 2$ (when $l_r = 1$) the algorithm has a relatively low perplexity, but in experiment $5, 6$ (when $l_r = 10$) the performance is very poor.

8. For exp 3.2 you trained a GRU. Was its performance as you expected and why?
Solution
As can be seen in Fig.8 and Fig.6 the performance of the GRU is better than RNN in all the cases studied. This behaviour is expected given that the GRU has more capacity and the RNNs only have a simple recurrent operations without any gates to control the flow of information among the cells.

9. In exp 3.3 you explored different hyperparameter settings in an attempt to improve the performance of the GRU. Were the validation/training curves as you expected for each setting? Comment on why. *Hint: For each hyperparameter setting, consider how the training and validation phases differ.*
Solution
From Fig.10 we can see the results. The results are as expected. When we increase the number of layers and the hidden layers we observe a considerable increase in the training time but the perplexity has a lower value. However, architectures with big hidden size are more susceptible to over-fitting as can be seen in the $ne = 4$

10. In exp 3.4 you trained a Transformer with various hyper-parameter settings. Given the recent high profile transformer based language models, are the results as you expected? Speculate as to why or why not.
Solution
In general terms, the transformer model, as expected had a good performance in all the cases studied, however its behavior was not much better than GRU. Perhaps working on a larger corpus or tuning the number of heads can give better results. it is remarkable that transformer seems to be susceptible to over-fitting. Additionally, it's worth say that two experiments of transformers had the shortest wall-clock time in the whole simulation bunch. This behaviour is reasonable given that it is a model that uses attention to boost the speed with which the model can be trained.

Problem 4

Detailed evaluation of trained models (25pts) For this problem, we will investigate properties of the trained models from Problem 3. Perform the following evaluations for the two models (one RNN and one GRU) for which the parameters were saved (indicated by the flag –save_best in the code).

1. For one minibatch of training data, compute the average gradient of the loss at the *final* time-step with respect to the hidden state at *each* time-step t: $\nabla_{\boldsymbol{h}_t}\mathcal{L}_T$. The norm of these gradients can be used to evaluate the propagation of gradients; a rapidly decreasing norm means that longer-term dependencies have less influence on the training signal, and can indicate **vanishing gradients**. Plot the Euclidian norm of $\nabla_{\boldsymbol{h}_t}\mathcal{L}_T$ as a function of t for the RNN and GRU architectures. Rescale the values of each curve to [0,1] so that you can compare both on one plot. Describe the results qualitatively, and provide an explanation for what you observe, discussing what the plots tell you about the gradient propagation in the different architectures.

Solution
The euclidian norm of gradients as function of the time steps for RNN and GRU can be observed in Fig.13. From Fig.13 we can clearly see the gradient vanishing problem in RNN, where the sequence 25 (of 35) already has a very small value of the gradient. This is problematic because small gradients means that the weights and biases of the initial layers will not be updated effectively with each training session. Since these initial layers are very important to define the context of the input data, it can affect the accuracy of the whole network. On the other hand, for the GRU model we can see that the gradients of the first sequences have considerable values.

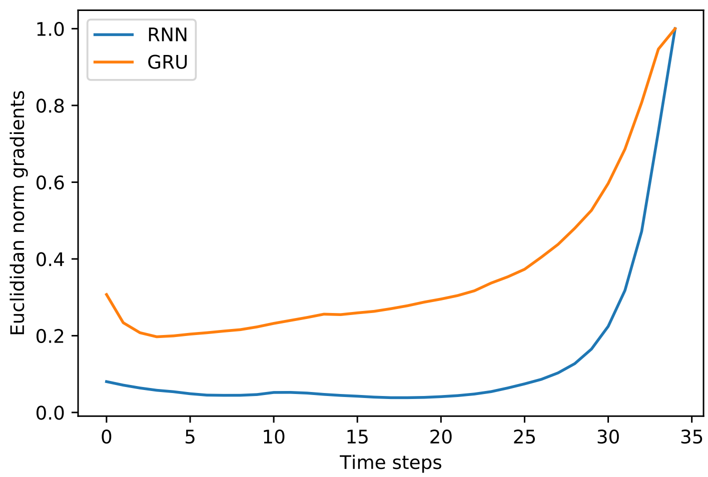

Figure 13: Euclidian norm of gradients as function of time steps for RNN and GRU

2. Generate samples from both the RNN and GRU models, by recursively making $\hat{\boldsymbol{x}}_{t+1} = \arg\max P(\boldsymbol{x}_{t+1}|\hat{\boldsymbol{x}}_1, ..., \hat{\boldsymbol{x}}_t)$.[3] Make sure to condition on the sampled $\hat{\boldsymbol{x}}_t$, *not* the ground truth. Produce 20 samples from both the RNN and GRU: 10 sequences of the same length as the training sequences, and 10 sequences of *twice* the length of the training sequences. Do you think that the generated sequence quality correlates with model validation perplexity? Justify your answer.

   Choose 3 "best", 3 "worst", and 3 that are "interesting". Put all 40 samples in an appendix to your report.

   Solution
   As previously discussed, GRU had less perplexity than RNN and this effect can be observed in the word sequences shown in the appendix. Generally speaking, RNN suffers from gradient vanishing, so the context of a sentence is easily lost in predicting words, resulting in meaningless sequences. On the contrary, GRU does not suffer with vanishing gradient and it is possible to form more reasonable sequences of words.

   - **Best:**
     – to introduce a new bid for the unk the company is expected to be acquired by the end of the year unk the company said it will sell its N N stake in navigation.

[3]It is possible to generate samples in the same manner from the Transformer, but the implementation is more involved, so you are not required to do so.

- airways plc a new york stock exchange eos the company said it has n't received any takeover bid eos the company said it would n't seek any bid for the company eos
- this year eos the company 's unk division was a unk of the nation 's largest steelmaker eos the company said it expects to report a loss of N million or N cents a

- **Worst:**
  - year eos the unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk
  - are n't expected to be fully operational eos the company is expected to be a unk of the unk and unk unk eos the unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk
  - the same time eos the unk is n't unk eos the unk is a unk unk eos the unk unk unk unk unk unk unk unk unk unk eos the unk unk unk unk unk unk unk unk eos the unk unk unk unk unk unk unk unk eos the unk unk unk unk unk unk unk unk eos the unk unk unk unk unk unk unk unk eos the
- **Interesting**
  - eos the company said it is n't unk to the unk eos the company said it is a unk unk eos the company said it is a unk unk eos the company said it will be a unk unk eos the company said it will be a unk unk eos the company said it is a unk unk eos the company said it will be a unk unk eos the
  - street 's unk eos the company said it is n't unk to the unk eos the company said it is a unk unk eos the company said it is a unk unk eos the company said it will be a unk unk eos the company said it will be a unk unk eos the company said it is a unk unk eos the company said it will be a unk
  - to be able to be a unk eos the company said it is a unk unk eos the company said it is a unk unk eos the company said it will be a unk unk eos the company said it will be a unk unk eos the company said it is a unk unk eos the company said it will be a unk unk eos the company said it will

Appendix

RNN 10 phrases of 35

- week 's unk eos the unk is a unk unk eos the unk unk unk unk unk unk unk unk unk eos the unk unk unk unk unk unk unk unk eos the unk
- co. said it will be a unk unk eos the company said it is a unk unk eos the company said it will be a unk unk eos the company said it will be a
- week 's unk eos the unk is a unk unk eos the unk unk unk unk unk unk unk unk unk eos the unk unk unk unk unk unk unk unk eos the unk

- of the company 's unk eos the company said it is a unk unk eos the company said it will be a unk unk eos the company said it will be a unk unk eos
- years ago eos the company 's unk unk unk unk unk unk unk unk eos the unk unk unk unk unk unk unk unk eos the unk unk unk unk unk unk unk unk eos
- of the company 's unk eos the company said it is a unk unk eos the company said it will be a unk unk eos the company said it will be a unk unk eos
- n't be n't be able to be a unk eos the company said it is a unk unk eos the company said it is a unk unk eos the company said it will be a
- co. said it will be a unk unk eos the company said it is a unk unk eos the company said it will be a unk unk eos the company said it will be a
- the same time eos the unk is n't unk eos the unk is a unk unk eos the unk unk unk unk unk unk unk unk unk eos the unk unk unk unk unk
- the same months eos the unk is a unk unk eos the unk unk unk unk unk unk unk unk unk eos the unk unk unk unk unk unk unk unk eos the unk

RNN 10 phrases of 70

- of the company 's unk eos the company said it is a unk unk eos the company said it will be a unk unk eos the company said it will be a unk unk eos the company said it is a unk unk eos the company said it will be a unk unk eos the company said it will be a unk unk eos the company said it is a
- eos the company 's unk unk unk unk unk unk unk unk eos the unk unk unk unk unk unk unk unk eos the unk unk unk unk unk unk unk unk eos the unk unk unk unk unk unk unk eos the unk unk unk unk unk unk unk unk unk eos the unk unk unk unk unk unk unk
- to acquire the company 's unk eos the company said it is a unk unk eos the company said it will be a unk unk eos the company said it will be a unk unk eos the company said it is a unk unk eos the company said it will be a unk unk eos the company said it will be a unk unk eos the company said it is
- N million shares eos the company said it will be a unk unk eos the company said it will be a unk unk eos the company said it is a unk unk eos the company said it will be a unk unk eos the company said it will be a unk unk eos the company said it is a unk unk eos the company said it will be a unk
- to be able to be a unk eos the company said it is a unk unk eos the company said it is a unk unk eos the company said it will be a unk unk eos the company said it will be a unk unk eos the company said it is a unk unk eos the company said it will be a unk unk eos the company said it will

- street 's unk eos the company said it is n't unk to the unk eos the company said it is a unk unk eos the company said it is a unk unk eos the company said it will be a unk unk eos the company said it will be a unk unk eos the company said it is a unk unk eos the company said it will be a unk
- the same time eos the unk is n't unk eos the unk is a unk unk eos the unk unk unk unk unk unk unk unk unk unk eos the unk unk unk unk unk unk unk unk eos the unk unk unk unk unk unk unk eos the unk unk unk unk unk unk unk unk eos the unk unk unk unk unk unk unk unk eos the
- they 're unk eos the unk is n't unk eos the unk is a unk unk eos the company said it is a unk unk eos the company said it is a unk unk eos the company said it will be a unk unk eos the company said it will be a unk unk eos the company said it is a unk unk eos the company said it will be
- eos the company said it is n't unk to the unk eos the company said it is a unk unk eos the company said it is a unk unk eos the company said it will be a unk unk eos the company said it will be a unk unk eos the company said it is a unk unk eos the company said it will be a unk unk eos the
- years ago eos the company 's unk unk unk unk unk unk unk unk eos the unk unk unk unk unk unk unk eos the unk unk unk unk unk unk unk eos the unk unk unk unk unk unk unk unk eos the unk unk unk unk unk unk unk unk eos the unk unk unk unk unk unk unk unk eos the unk unk unk unk

GRU 10 phrases of 35

- to introduce a new bid for the unk the company is expected to be acquired by the end of the year unk the company said it will sell its N N stake in navigation.
- airways plc a new york stock exchange eos the company said it has n't received any takeover bid eos the company said it would n't seek any bid for the company eos
- rates on the treasury 's benchmark 30-year bond eos the yield on the treasury 's benchmark 30-year bond ended at N N to yield N N compared with N N to yield N N eos
- unk and unk unk eos the unk of the unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk
- banker unk unk & co. a new york firm eos the new york stock exchange has been unk by the new york stock exchange eos the big board is a unk of the nation 's
- of commons eos the unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk
- of the N N of the total of N billion eos the unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk
- a share of N cents a share eos the company also said it is n't interested in the company eos the company said it has n't yet been able to sell its shares in the

- this year eos the company 's unk division was a unk of the nation 's largest steelmaker eos the company said it expects to report a loss of N million or N cents a
- a share of N cents a share eos the company also said it is n't interested in the company eos the company said it has n't yet been able to sell its shares in the

GRU 10 phrases of 70

- to introduce a new bid for the unk eos the company is expected to be acquired by the end of the year eos the company said it will sell its N N stake in navigation mixte eos the company said it will sell its N N stake in navigation mixte eos paribas said it will sell its N N stake in navigation mixte eos paribas said it will sell its
- rates eos the average seven-day compound yield of N N and N N of the year ended oct. N eos the average rate of N N was up N N from N N eos the average rate of N N was down N N from N N eos the average rate of N N was down N N from N N eos the average rate of N N was down
- are n't expected to be fully operational eos the company is expected to be a unk of the unk and unk unk eos the unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk
- to make sure it is a unk eos but the unk of the unk is unk eos the unk of the unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk
- be able to buy the unk eos the unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk
- other things eos the unk of the unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk
- year eos the unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk
- rates in the past N days eos the average rate of N N and N N respectively eos the average rate of N N was up N N from N N in august eos the N N N notes due N fell N to N N to yield N N eos the N N N notes due N fell N to N N to yield N N eos the latest
- are n't expected to be fully operational eos the company is expected to be a unk of the unk and unk unk eos the unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk

unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk unk

- evasion eos the bill is subject to a bill to be paid by a federal reserve court eos the bill is subject to a bill to raise the minimum wage for the first six months of the year eos the treasury 's benchmark 30-year bond ended at N N to yield N N compared with N N to yield N N eos the treasury 's benchmark 30-year bond ended at