IFT 6135 - Homework 2

Antoine Chehire, Philippe Marchandise, Adel Nabli

25/03/2019

Link of the Github where the code used is stored: `https://github.com/achehire/Deep_1_AC_AN_PW`

1 Problem 1

```
class RNN(nn.Module): # Implement a stacked vanilla RNN with Tanh nonlinearities.
    def __init__(self, emb_size, hidden_size, seq_len, batch_size, vocab_size,
                 num_layers, dp_keep_prob):

        self.emb_size = emb_size
        self.hidden_size = hidden_size
        self.seq_len = seq_len
        self.batch_size = batch_size
        self.vocab_size = vocab_size
        self.num_layers = num_layers
        self.dp_keep_proba = dp_keep_prob
        super(RNN, self).__init__()

        # Embeddings
        self.embeddings = nn.Embedding(vocab_size, emb_size)

        # Output
        self.lin_o = nn.Linear(hidden_size, vocab_size, bias=True)

        # Hidden weights
        for layer in range(num_layers):
            layer_input_size = emb_size if layer == 0 else hidden_size

            lin_x = nn.Linear(layer_input_size, hidden_size, bias=True)
            lin_h = nn.Linear(hidden_size, hidden_size, bias=False)

            setattr(self, 'lin_x_{}'.format(layer), lin_x)
            setattr(self, 'lin_h_{}'.format(layer), lin_h)

        # Dropout
        self.drop = nn.Dropout(p=1-dp_keep_prob)

        self.init_weights()
        self.init_hidden()

    def init_weights(self):

        init.uniform_(self.embeddings.weight, -0.1, 0.1)
        init.uniform_(self.lin_o.weight, -0.1, 0.1)
        init.zeros_(self.lin_o.bias)

        stdv = 1.0 / math.sqrt(self.hidden_size)
        for layer in range(self.num_layers):
            init.uniform_(getattr(self, "lin_x_{}".format(layer)).weight, -stdv, stdv)
            init.uniform_(getattr(self, "lin_x_{}".format(layer)).bias, -stdv, stdv)
            init.uniform_(getattr(self, "lin_h_{}".format(layer)).weight, -stdv, stdv)

    def init_hidden(self):
        return nn.Parameter(torch.zeros(self.num_layers, self.batch_size, self.hidden_size),
                            False)

    def forward(self, inputs, hidden):

        logits = []

        for words_t in inputs:
            x_t = self.drop(self.embeddings(words_t))

            for layer in range(self.num_layers):
                h_t = hidden[layer].clone()
                h_t = torch.tanh(getattr(self, "lin_x_{}".format(layer))(x_t)+
                                 getattr(self, "lin_h_{}".format(layer))(h_t))
                x_t = self.drop(h_t)
                hidden[layer] = h_t

            out = self.lin_o(x_t)
            logits.append(out)

        logits = torch.stack(logits)
        return logits.view(self.seq_len, self.batch_size, self.vocab_size), hidden
```

2 Problem 2

```
class GRU(nn.Module): # Implement a stacked GRU RNN

    def __init__(self, emb_size, hidden_size, seq_len, batch_size, vocab_size,
                 num_layers, dp_keep_prob):
        self.emb_size = emb_size
        self.hidden_size = hidden_size
        self.seq_len = seq_len
        self.batch_size = batch_size
        self.vocab_size = vocab_size
        self.num_layers = num_layers
        self.dp_keep_proba = dp_keep_prob
        super(GRU, self).__init__()

        # Embeddings
        self.embeddings = nn.Embedding(vocab_size, emb_size)

        # Output
        self.lin_o = nn.Linear(hidden_size, vocab_size, bias=True)

        # Hidden weights
        for layer in range(num_layers):
            layer_input_size = emb_size if layer == 0 else hidden_size

            lin_rx = nn.Linear(layer_input_size, hidden_size, bias=True)
            lin_rh = nn.Linear(hidden_size, hidden_size, bias=False)
            lin_zx = nn.Linear(layer_input_size, hidden_size, bias=True)
            lin_zh = nn.Linear(hidden_size, hidden_size, bias=False)
            lin_hx = nn.Linear(layer_input_size, hidden_size, bias=True)
            lin_hh = nn.Linear(hidden_size, hidden_size, bias=False)

            setattr(self, 'lin_rx_{}'.format(layer), lin_rx)
            setattr(self, 'lin_rh_{}'.format(layer), lin_rh)
            setattr(self, 'lin_zx_{}'.format(layer), lin_zx)
            setattr(self, 'lin_zh_{}'.format(layer), lin_zh)
            setattr(self, 'lin_hx_{}'.format(layer), lin_hx)
            setattr(self, 'lin_hh_{}'.format(layer), lin_hh)

        # Dropout
        self.drop = nn.Dropout(p=1-dp_keep_prob)

        self.init_weights()
        self.init_hidden()

    def init_weights(self):
        init.uniform_(self.embeddings.weight, -0.1, 0.1)
        init.uniform_(self.lin_o.weight, -0.1, 0.1)
        init.zeros_(self.lin_o.bias)

        stdv = 1.0 / math.sqrt(self.hidden_size)
        for layer in range(self.num_layers):
            init.uniform_(getattr(self, "lin_rx_{}".format(layer)).weight, -stdv, stdv)
            init.uniform_(getattr(self, "lin_rx_{}".format(layer)).bias, -stdv, stdv)
            init.uniform_(getattr(self, "lin_rh_{}".format(layer)).weight, -stdv, stdv)
            init.uniform_(getattr(self, "lin_zx_{}".format(layer)).weight, -stdv, stdv)
            init.uniform_(getattr(self, "lin_zx_{}".format(layer)).bias, -stdv, stdv)
            init.uniform_(getattr(self, "lin_zh_{}".format(layer)).weight, -stdv, stdv)
            init.uniform_(getattr(self, "lin_hx_{}".format(layer)).weight, -stdv, stdv)
            init.uniform_(getattr(self, "lin_hx_{}".format(layer)).bias, -stdv, stdv)
            init.uniform_(getattr(self, "lin_hh_{}".format(layer)).weight, -stdv, stdv)

    def init_hidden(self):
        return nn.Parameter(torch.zeros(self.num_layers, self.batch_size, self.hidden_size),
                            False)

    def forward(self, inputs, hidden):

        logits = []

        for words_t in inputs:

            x_t = self.drop(self.embeddings(words_t))

            for layer in range(self.num_layers):
                h_t = hidden[layer].clone()

                r_t = torch.sigmoid(getattr(self, "lin_rx_{}".format(layer))(x_t) +
                                    getattr(self, "lin_rh_{}".format(layer))(h_t))
                z_t = torch.sigmoid(getattr(self, "lin_zx_{}".format(layer))(x_t) +
                                    getattr(self, "lin_zh_{}".format(layer))(h_t))
                h_tilde = torch.tanh(getattr(self, "lin_hx_{}".format(layer))(x_t) +
                                     getattr(self, "lin_hh_{}".format(layer))(h_t*r_t))
                h_t = (1-z_t)*h_t + z_t*h_tilde
                x_t = self.drop(h_t)

                hidden[layer] = h_t

            out = self.lin_o(x_t)
            logits.append(out)

        logits = torch.stack(logits)

        return logits.view(self.seq_len, self.batch_size, self.vocab_size), hidden
```

3 Problem 3

```
class MultiHeadedAttention(nn.Module):
    def __init__(self, n_heads, n_units, dropout=0.1):
        super(MultiHeadedAttention, self).__init__()
        # This sets the size of the keys, values, and queries (self.d_k) to all
        # be equal to the number of output units divided by the number of heads.
        self.d_k = n_units // n_heads
        # This requires the number of n_heads to evenly divide n_units.
        assert n_units % n_heads == 0
        self.n_units = n_units
        self.n_heads = n_heads

        self.lin_q = clones(nn.Linear(self.n_units, self.d_k, bias=True), n_heads)
        self.lin_k = clones(nn.Linear(self.n_units, self.d_k, bias=True), n_heads)
        self.lin_v = clones(nn.Linear(self.n_units, self.d_k, bias=True), n_heads)
        self.lin_o = nn.Linear(self.n_units, self.n_units, bias=True)
        self.drop = nn.Dropout(dropout)

        # init:
        k = 1.0/math.sqrt(n_units)

        for head in range(n_heads):

            init.uniform_(self.lin_q[head].weight, -k, k)
            init.uniform_(self.lin_k[head].weight, -k, k)
            init.uniform_(self.lin_v[head].weight, -k, k)
            init.uniform_(self.lin_q[head].bias, -k, k)
            init.uniform_(self.lin_k[head].bias, -k, k)
            init.uniform_(self.lin_v[head].bias, -k, k)

        init.uniform_(self.lin_o.bias, -k, k)
        init.uniform_(self.lin_o.weight, -k, k)

    def forward(self, query, key, value, mask=None):
        H = []
        soft = torch.nn.Softmax(dim=-1)

        for head in range(self.n_heads):

            a_q = self.lin_q[head](query)
            a_k = self.lin_k[head](key)
            a_v = self.lin_v[head](value)
            x = torch.matmul(a_q, a_k.transpose(1,2)) / math.sqrt(self.d_k)
            mask = mask.float() # the given mask is Byte
            a_i = soft(x*mask - 10**9*(1-mask))
            h_i = torch.matmul(self.drop(a_i), a_v)
            H.append(h_i)

        H = torch.cat(H, dim=2)
        A = self.lin_o(H)

        return A
```

4 Problem 4

4.1 Figures and tables

1. We report below the 36 learning curves. For each of the 18 experiments, we drew the learning curves in function of both the epochs and the wall clock time.

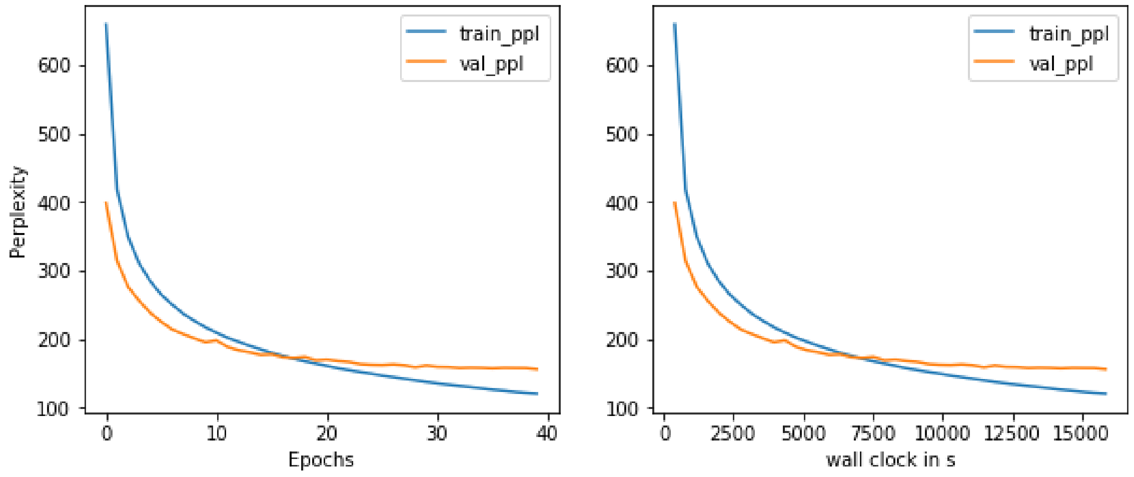


Figure r1: Learning curve of the RNN using Adam

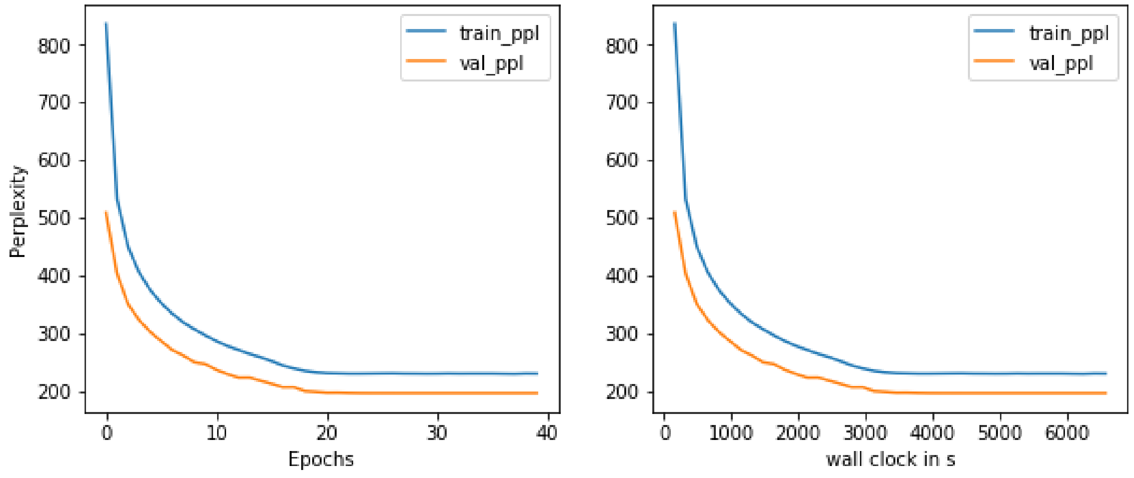


Figure r2: Learning curve of the RNN using sgd schedule

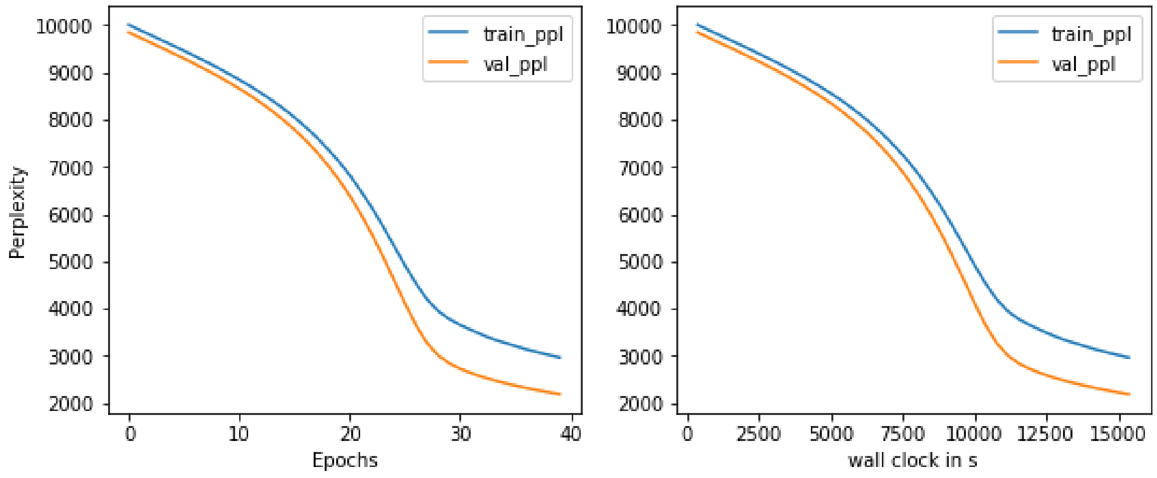


Figure r3: Learning curve of the RNN using sgd

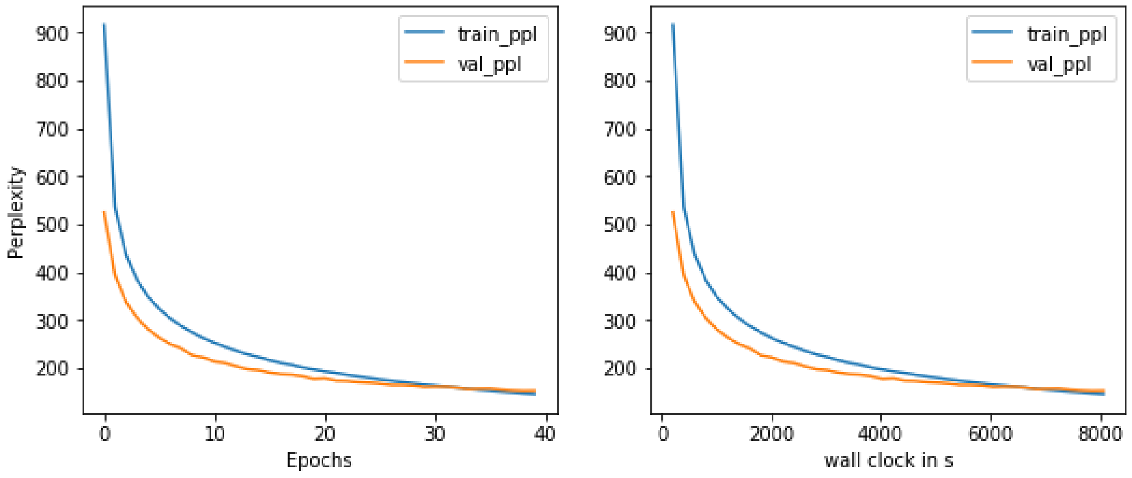


Figure r4: Learning curve of the RNN using adam with higher hidden size

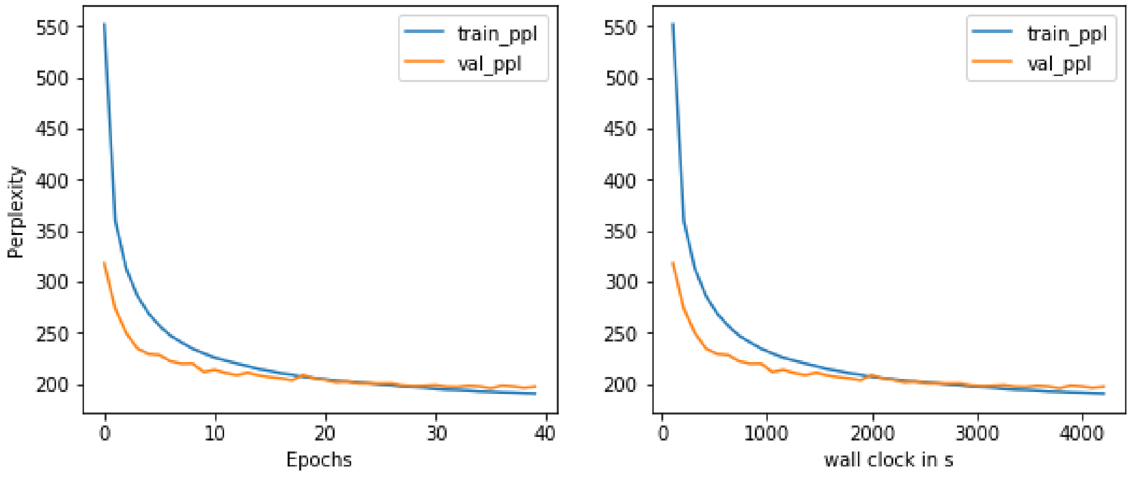


Figure r5: Learning curve of the RNN using adam with lower hidden size and a third layer

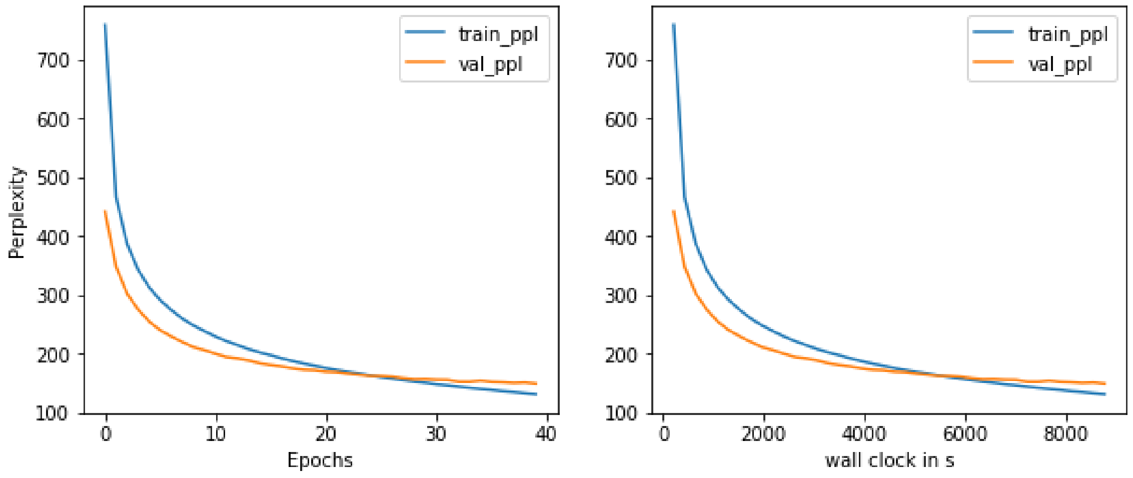


Figure r6: Learning curve of the RNN using adam with larger batch size

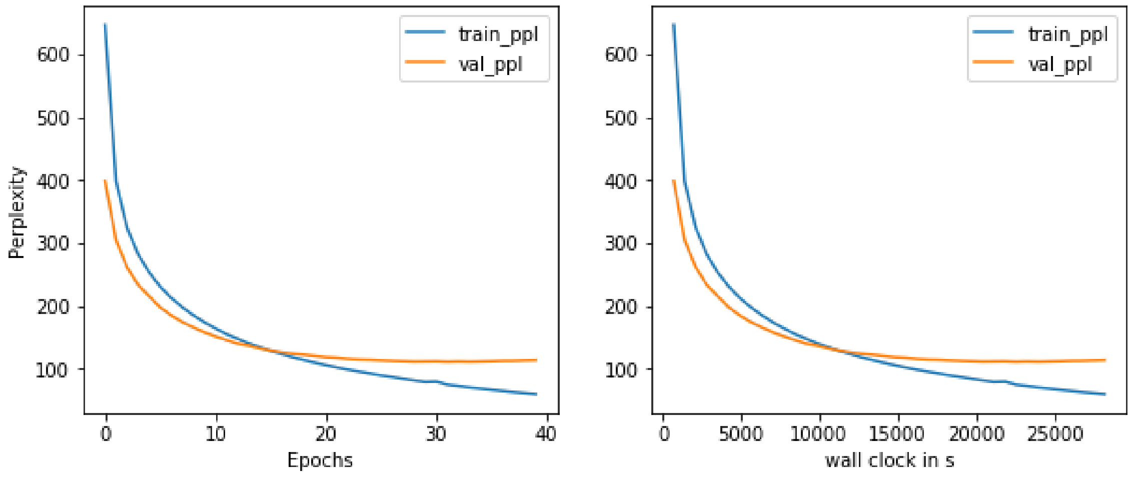


Figure g1: Learning curve of the GRU using adam

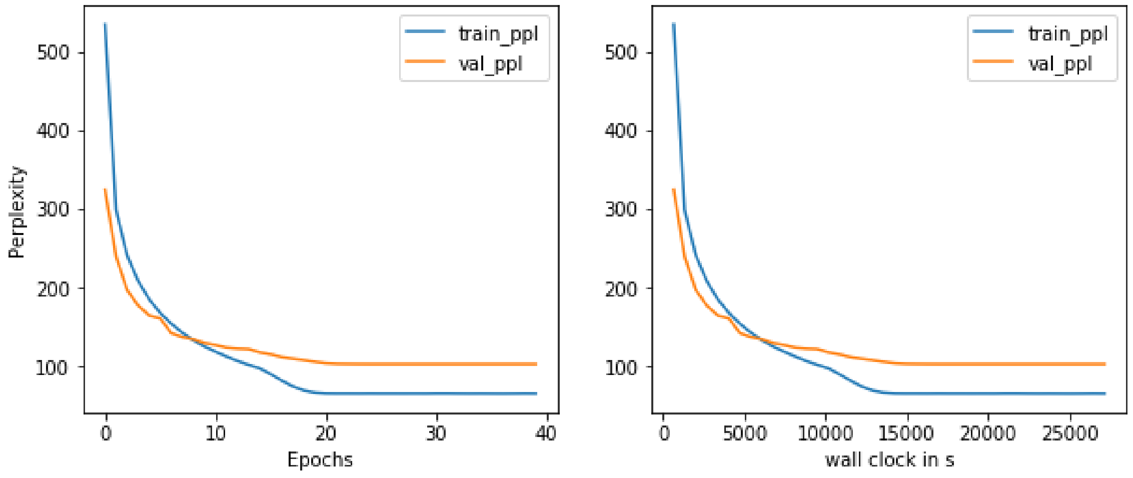


Figure g2: Learning curve of the GRU using sgd schedule

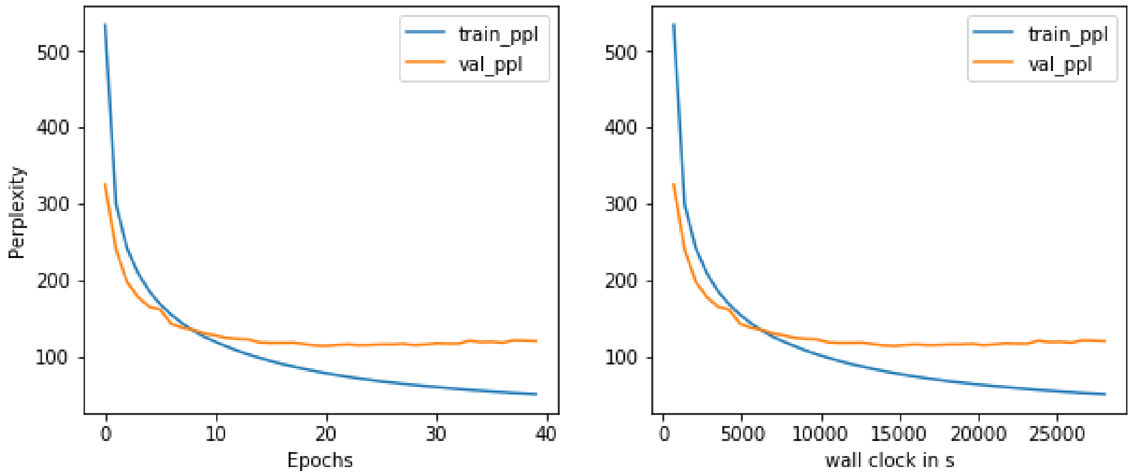


Figure g3: Learning curve of the GRU using sgd

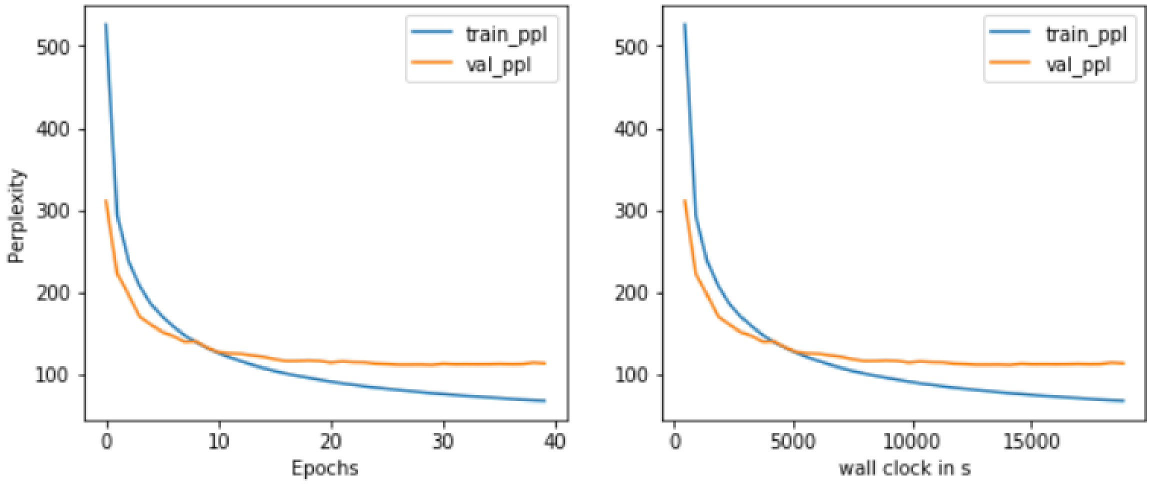


Figure g4: Learning curve of the GRU using sgd with smaller hidden size

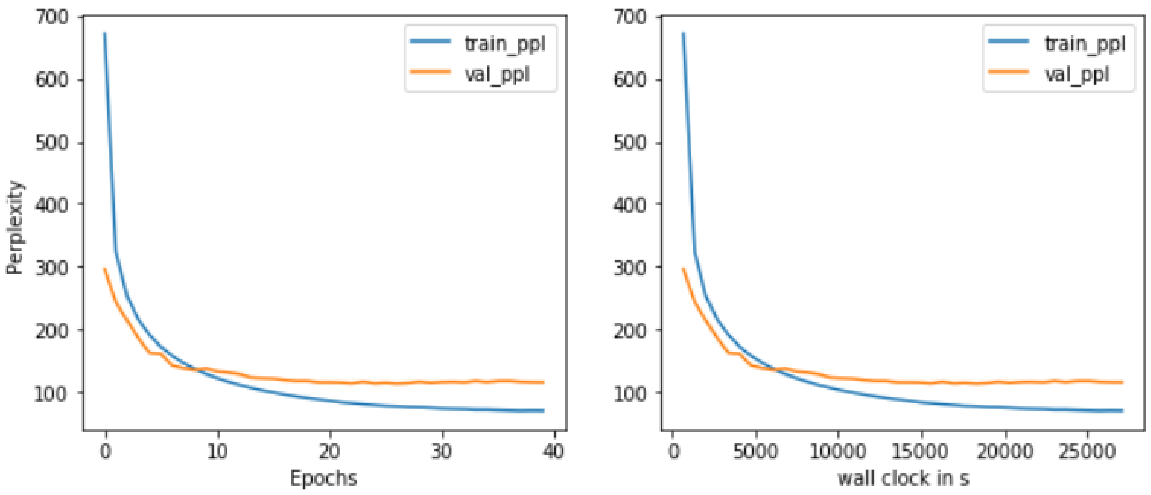


Figure g5: Learning curve of the GRU using sgd with larger learning rate

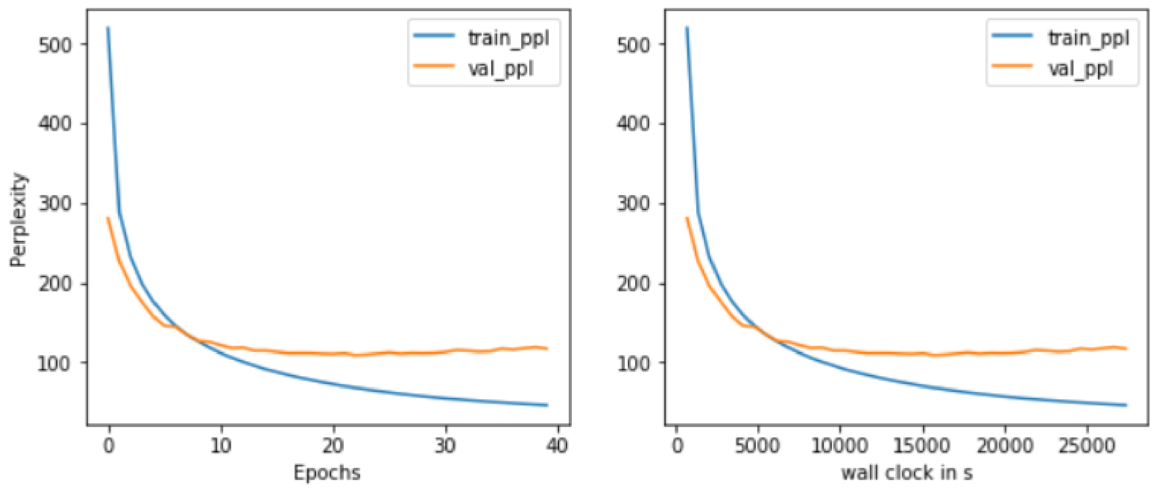


Figure g6: Learning curve of the GRU using sgd with larger embedding size

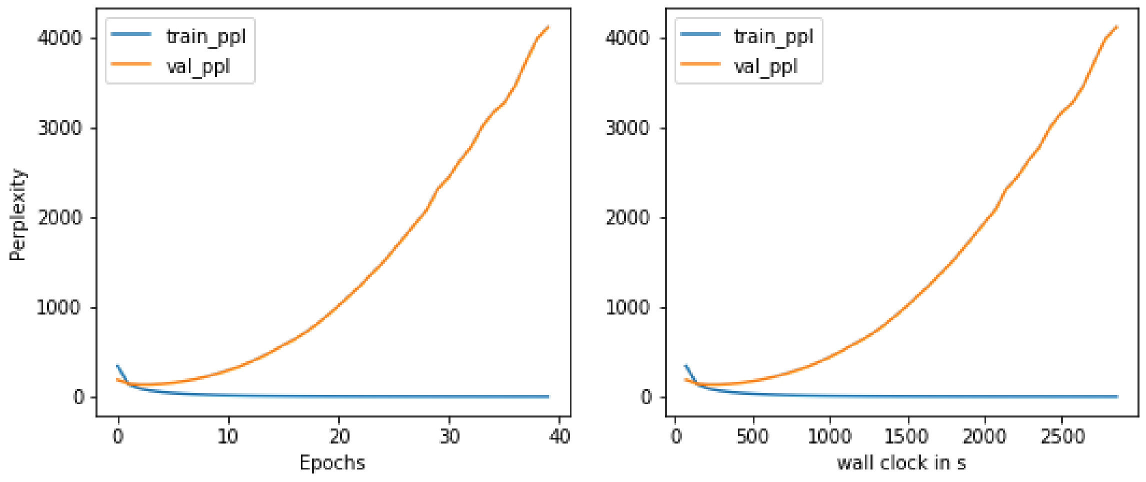


Figure t1: Learning curve of the Transformer using adam

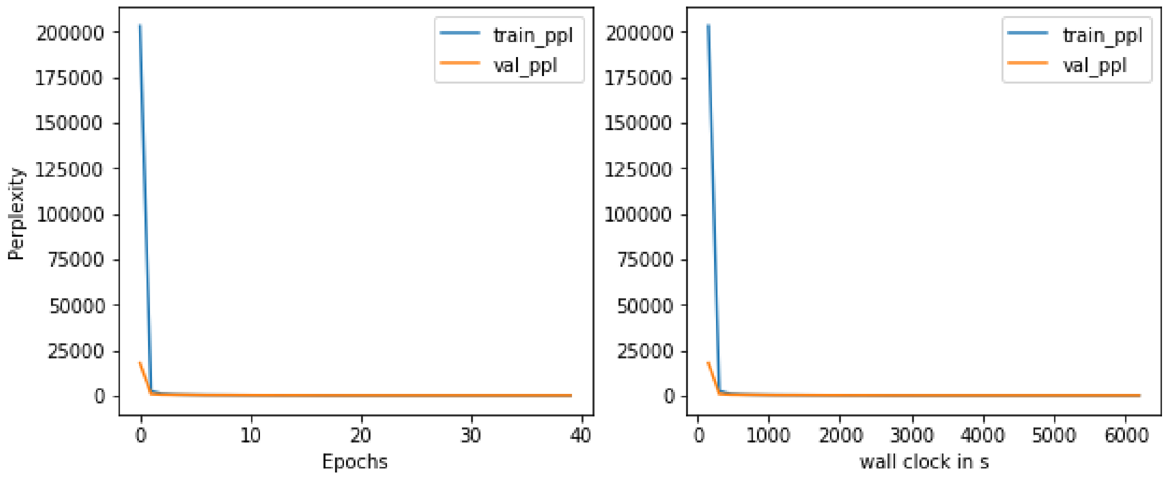


Figure t2: Learning curve of the Transformer using sgd schedule

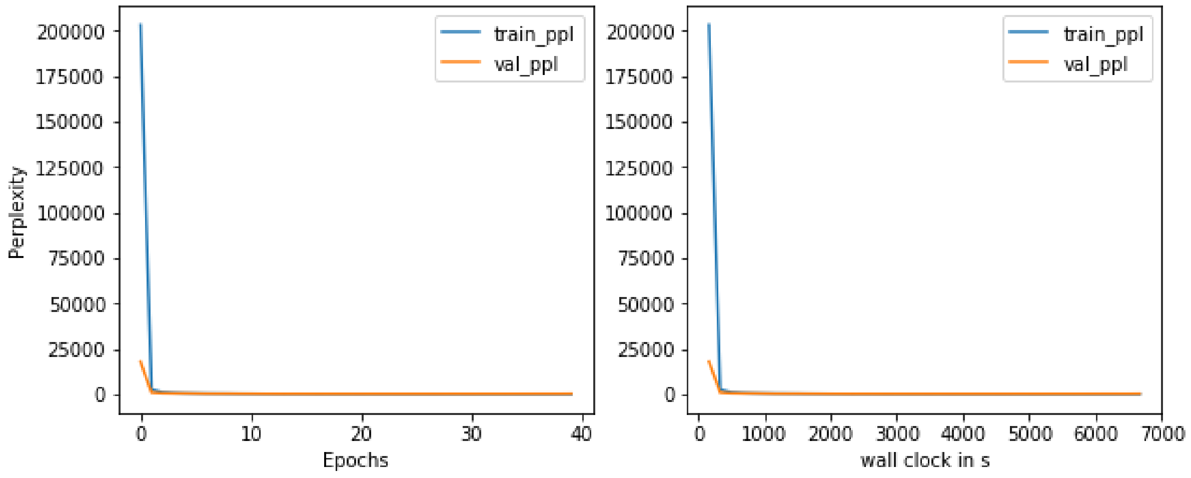


Figure t3: Learning curve of the Transformer using sgd

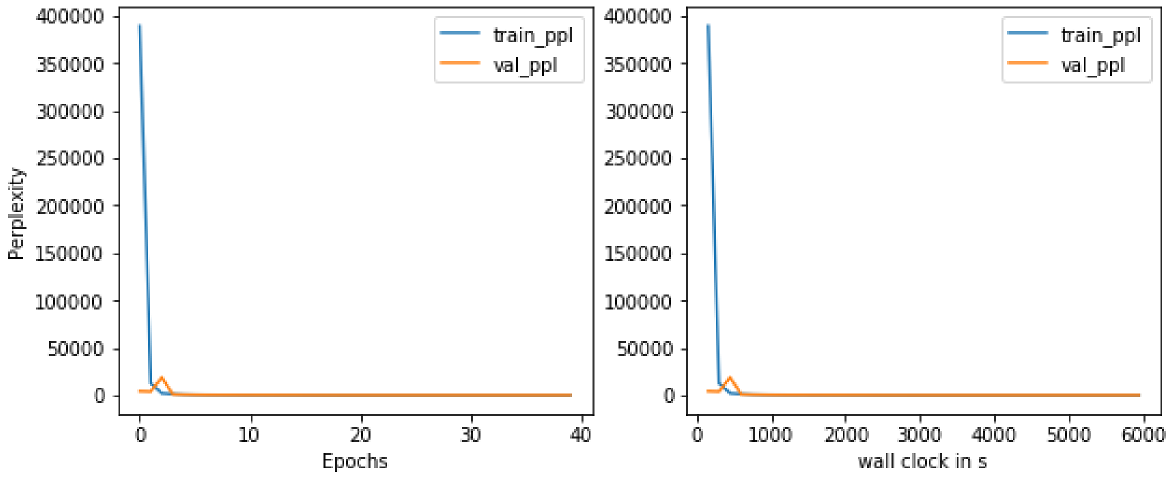


Figure t4: Learning curve of the Transformer using sgd schedule with higher sequence length

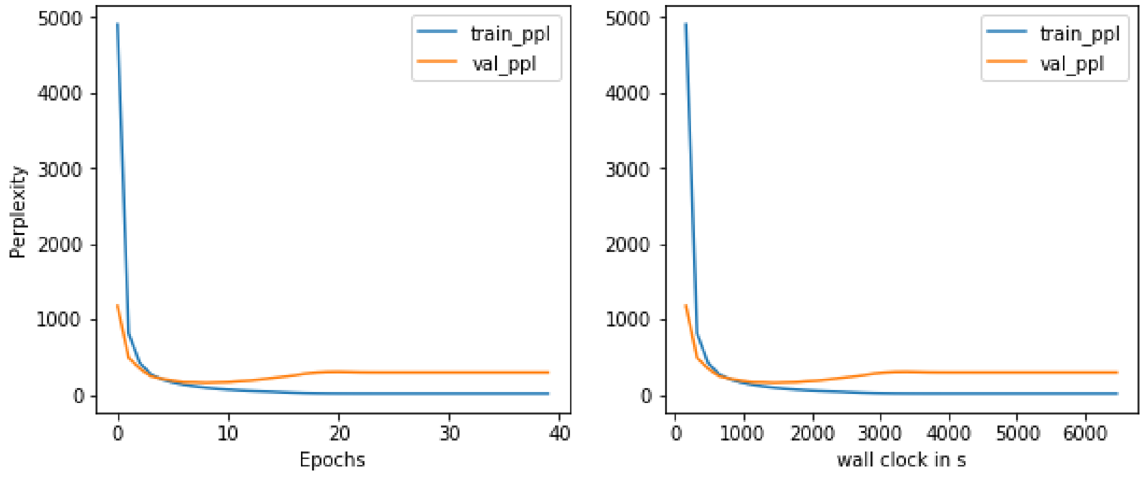


Figure t5: Learning curve of the Transformer using sgd schedule with lower learning rate

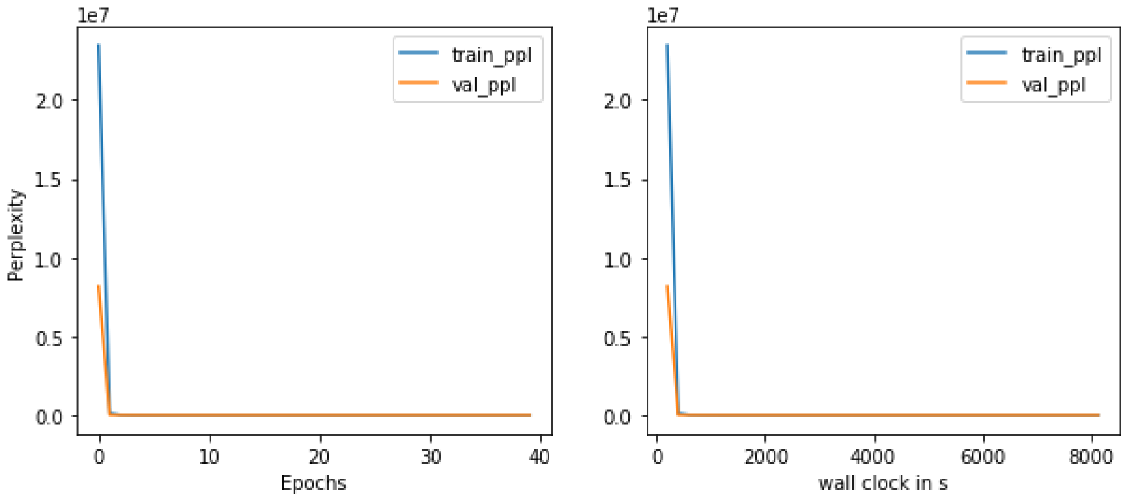
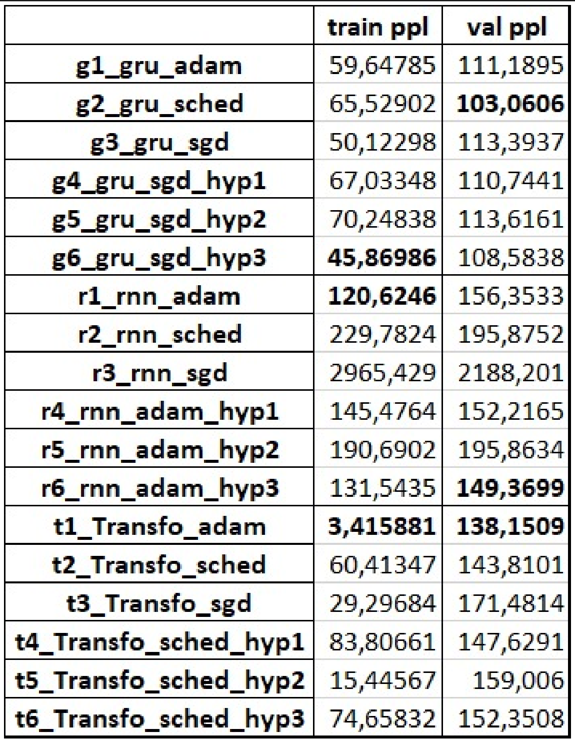


Figure t6: Learning curve of the Transformer using sgd schedule with greater hidden size

2. In order to compare all these experiments, we reported the best training and validation perplexities in the table below:

Figure 1: Table containing the best results for each experiment



	train ppl	**val ppl**
g1_gru_adam	59,64785	111,1895
g2_gru_sched	65,52902	**103,0606**
g3_gru_sgd	50,12298	113,3937
g4_gru_sgd_hyp1	67,03348	110,7441
g5_gru_sgd_hyp2	70,24838	113,6161
g6_gru_sgd_hyp3	**45,86986**	108,5838
r1_rnn_adam	**120,6246**	156,3533
r2_rnn_sched	229,7824	195,8752
r3_rnn_sgd	2965,429	2188,201
r4_rnn_adam_hyp1	145,4764	152,2165
r5_rnn_adam_hyp2	190,6902	195,8634
r6_rnn_adam_hyp3	131,5435	**149,3699**
t1_Transfo_adam	**3,415881**	**138,1509**
t2_Transfo_sched	60,41347	143,8101
t3_Transfo_sgd	29,29684	171,4814
t4_Transfo_sched_hyp1	83,80661	147,6291
t5_Transfo_sched_hyp2	15,44567	159,006
t6_Transfo_sched_hyp3	74,65832	152,3508

3. We report the parameters we chose for each experiment:

```
g1: model=GRU; optimizer=ADAM; initial lr=0.0001; batch size=20; seq len=35;
    hidden size=1500; num layers=2; dp keep prob=0.35
g2: model=GRU; optimizer=SGD LR SCHEDULE; initial lr=10; batch size=20; seq len=35;
    hidden size=1500; num layers=2; dp keep prob=0.35
g3: model=GRU; optimizer=SGD; initial lr=10; batch size=20; seq len=35;
    hidden size=1500; num layers=2; dp keep prob=0.35
g4: model=GRU; optimizer=SGD; initial lr=10; batch size=20; seq len=35;
    hidden size=1000; num layers=2; dp keep prob=0.35
g5: model=GRU; optimizer=SGD; initial lr=15; batch size=20; seq len=35;
```

```
     hidden size=1500; num layers=2; dp keep prob=0.35
g6: model=GRU; optimizer=SGD; initial lr=10; batch size=20; seq len=35;
     hidden size=1500; num layers=2; dp keep prob=0.35; emb size=300
r1: model=RNN; optimizer=ADAM; initial lr=0.0001; batch size=20; seq len=35;
     hidden size=1500; num layers=2; dp keep prob=0.35
r2: model=RNN; optimizer=SGD LR SCHEDULE; initial lr=1; batch size=20; seq len=35;
     hidden size=512; num layers=2; dp keep prob=0.35
r3: model=RNN; optimizer=SGD; initial lr=0.0001; batch size=20; seq len=35;
     hidden size=1500; num layers=2; dp keep prob=0.35
r4: model=RNN; optimizer=ADAM; initial lr=0.0001; batch size=100; seq len=35;
     hidden size=2000; num layers=2; dp keep prob=0.35
r5: model=RNN; optimizer=ADAM; initial lr=0.001; batch size=40; seq len=35;
     hidden size=512; num layers=3; dp keep prob=0.35
r6: model=RNN; optimizer=ADAM; initial lr=0.0001; batch size=40; seq len=35;
     hidden size=1500; num layers=2; dp keep prob=0.35
t1: model=TRANSFORMER; optimizer=ADAM; initial lr=0.001; batch size=128; seq len=35;
     hidden size=512; num layers=2; dp keep prob=.9
t2: model=TRANSFORMER; optimizer=SGD LR SCHEDULE; initial lr=20; batch size=128; seq len=35;
     hidden size=512; num layers=6; dp keep prob=0.9
t3: model=TRANSFORMER; optimizer=SGD; initial lr=20; batch size=128; seq len=35;
     hidden size=512; num layers=6; dp keep prob=.9
t4: model=TRANSFORMER; optimizer=SGD LR SCHEDULE; initial lr=20; batch size=128; seq len=42;
     hidden size=512; num layers=6; dp keep prob=0.9
t5: model=TRANSFORMER; optimizer=SGD LR SCHEDULE; initial lr=10; batch size=128; seq len=35;
     hidden size=512; num layers=6; dp keep prob=0.9
t6: model=TRANSFORMER; optimizer=SGD LR SCHEDULE; initial lr=20; batch size=128; seq len=35;
     hidden size=1024; num layers=6; dp keep prob=0.9
```

4. As the results reported in the table don't give enough information on how the model behaves when the number of epochs increases, we also drew the curves obtained for each optimizer. As the transformer models have a very high initial perplexity, we also plotted the perplexity from epoch 4 on wards (ie after 5 epochs due to the notation):

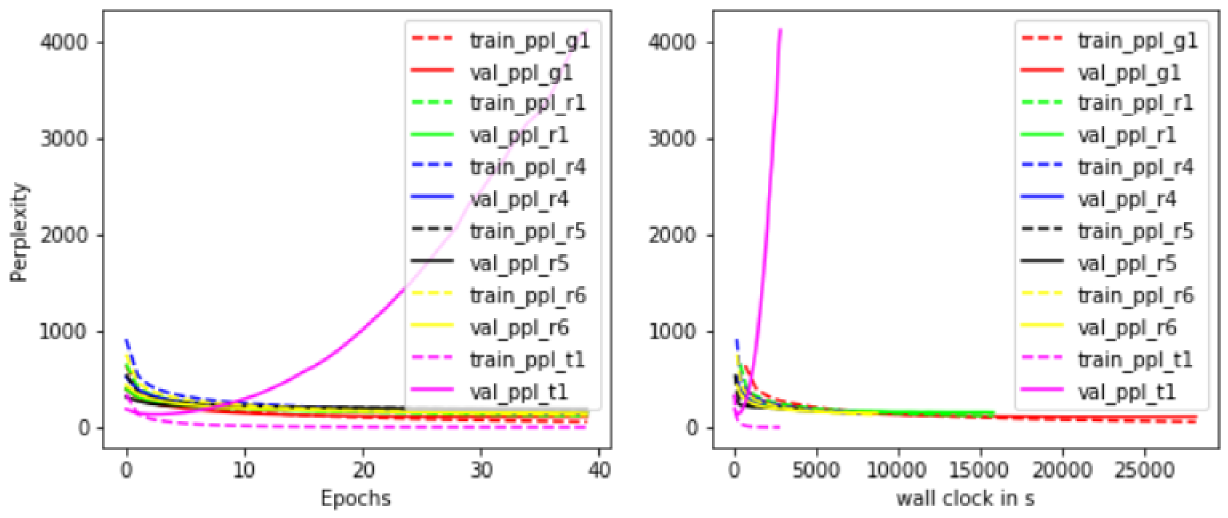


Figure ADAM A: Learning curves for ADAM

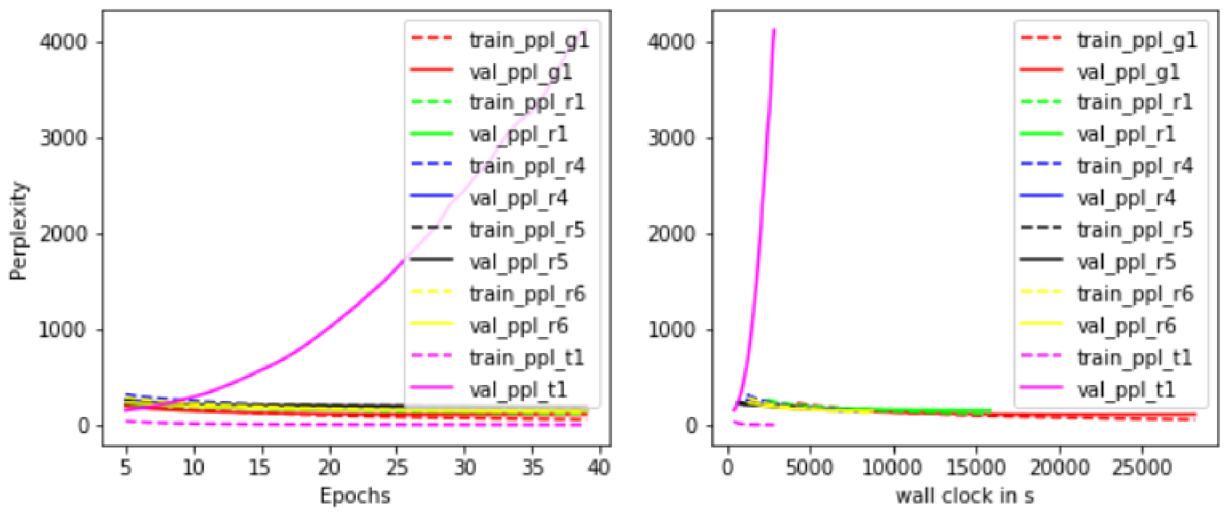


Figure ADAM B: Learning curves for ADAM after 5 epochs

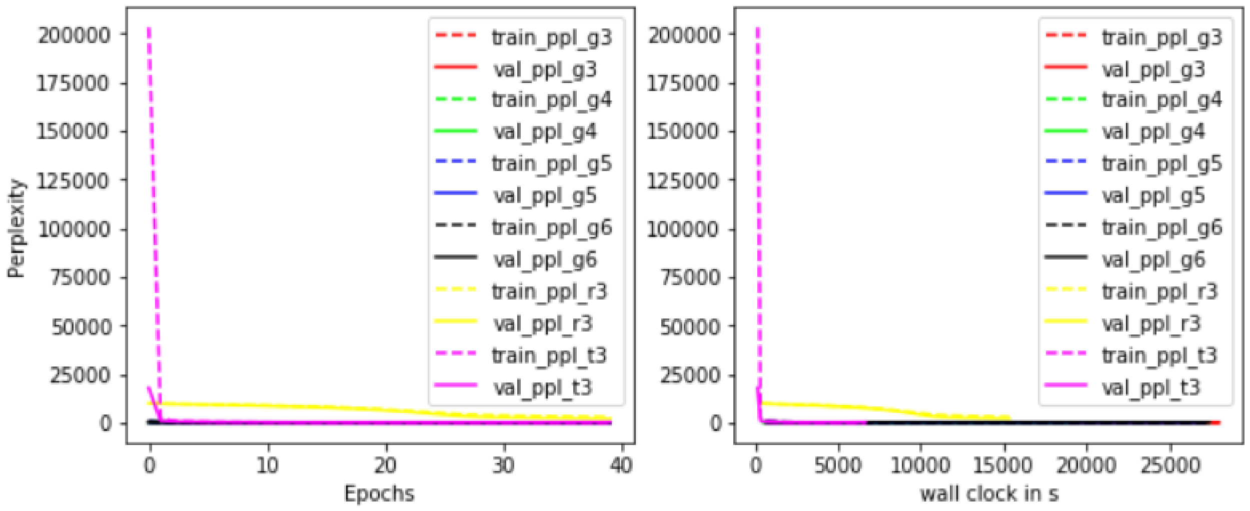


Figure SGD A: Learning curves for SGD

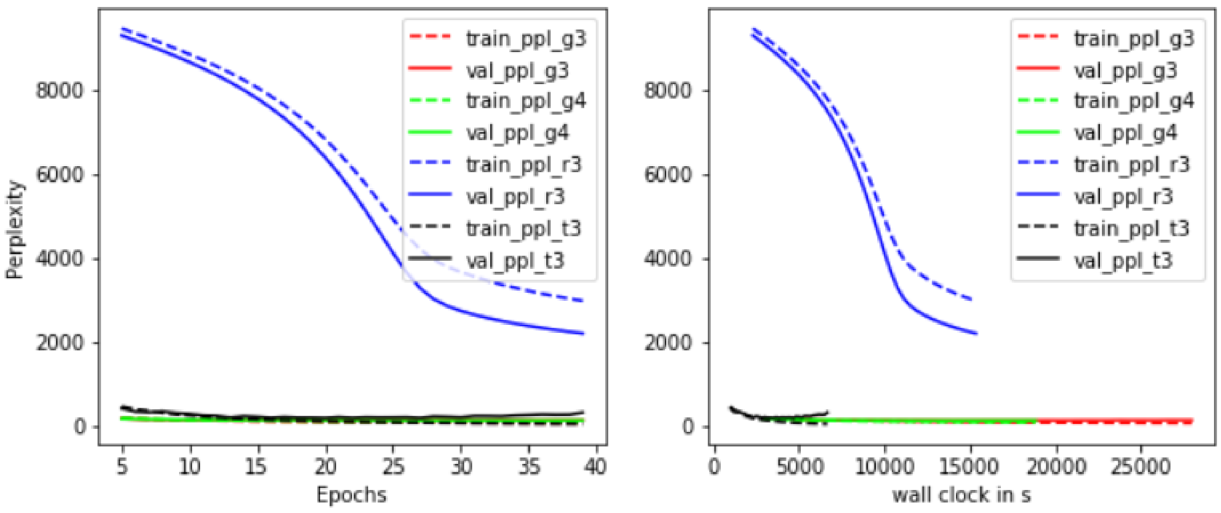


Figure SGD B: Learning curves for SGD after 5 epochs

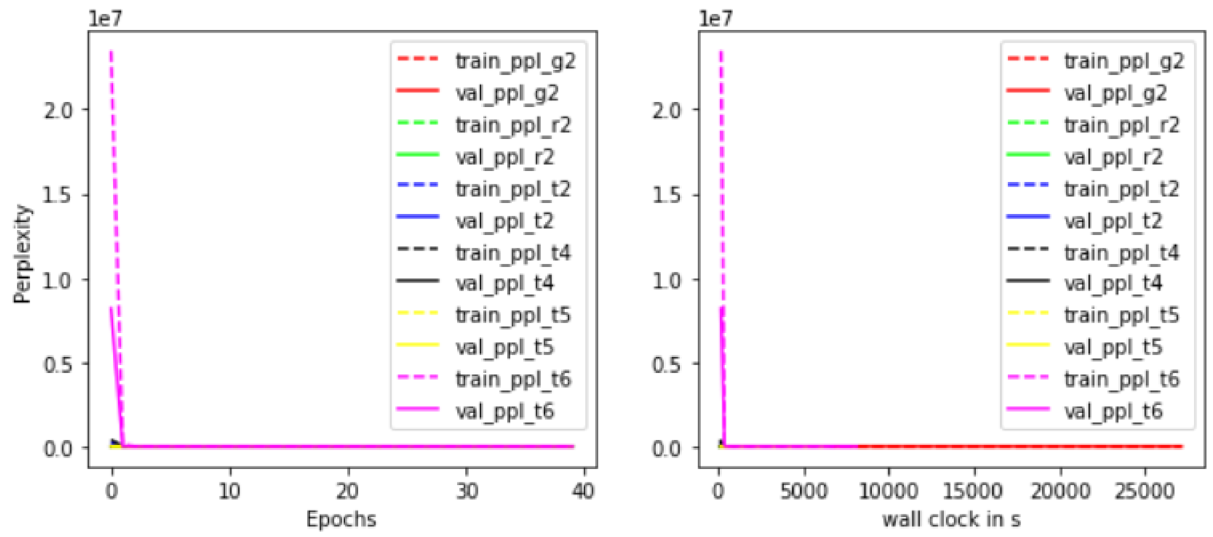


Figure SCHED A: Learning curves for SGD LR SCHEDULE

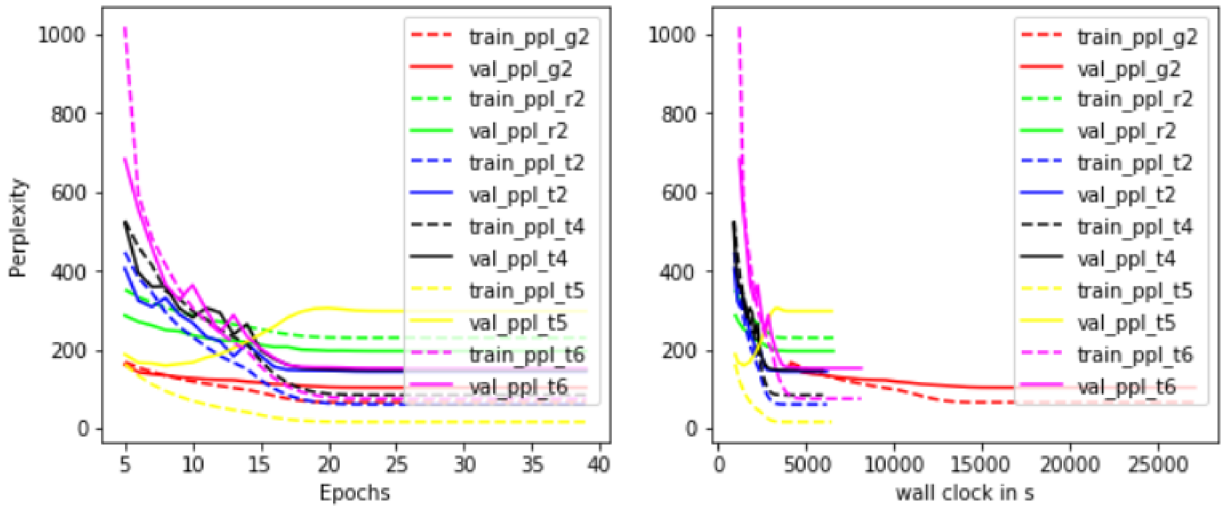


Figure SCHED B: Learning curves for SGD LR SCHEDULE after 5 epochs

5. We also plotted the curves for each model. In some cases, we felt it necessary to plot another figure in order to analyze the plots properly:

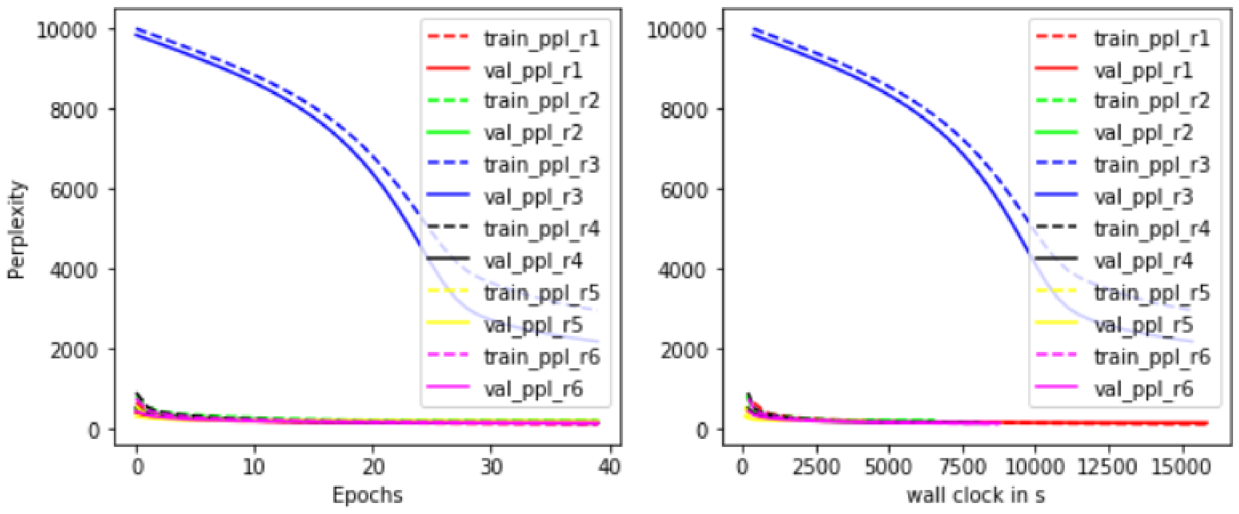


Figure RNN: Learning curves for the RNN

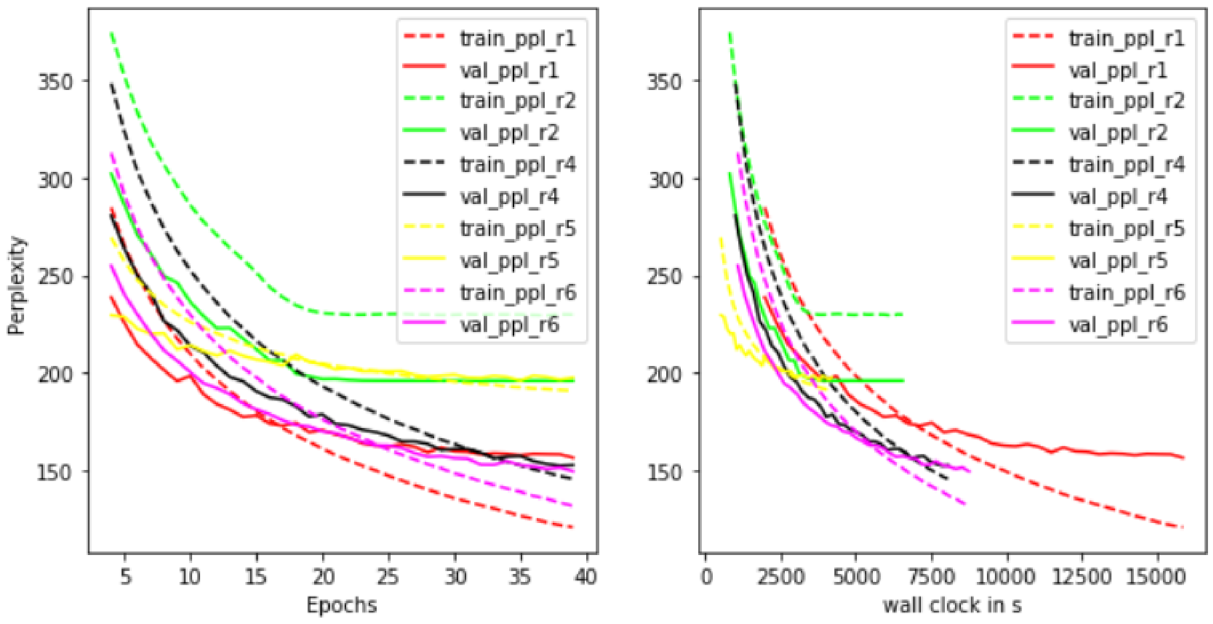


Figure RNN B: Learning curves for the RNN after 5 epochs and without **r3**

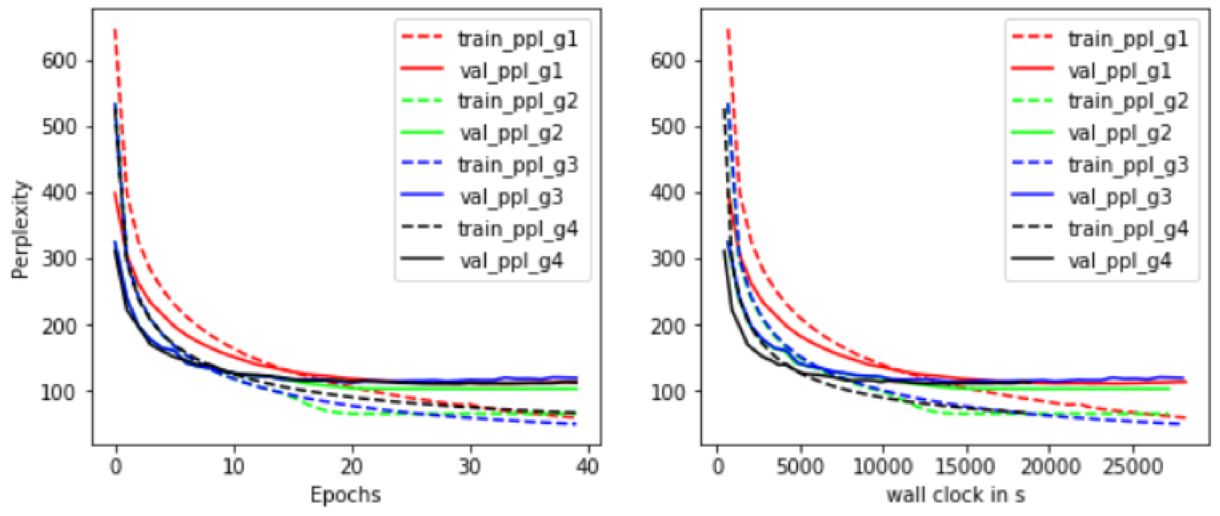


Figure GRU: Learning curves for the GRU

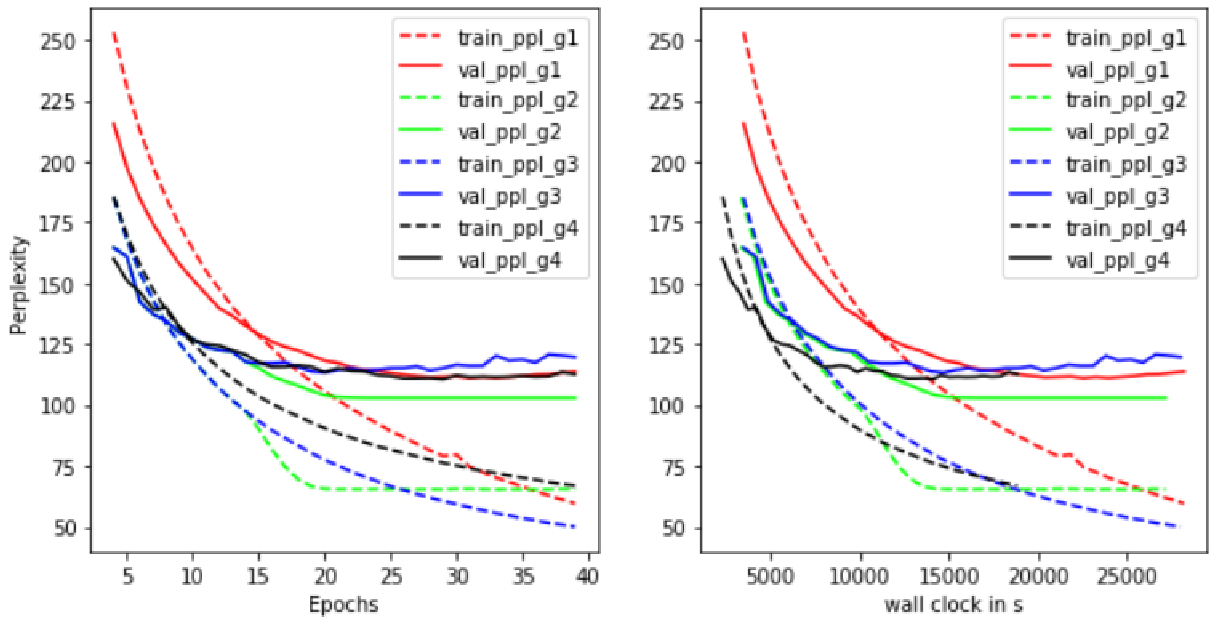


Figure GRU B: Learning curves for the GRU after 5 epochs

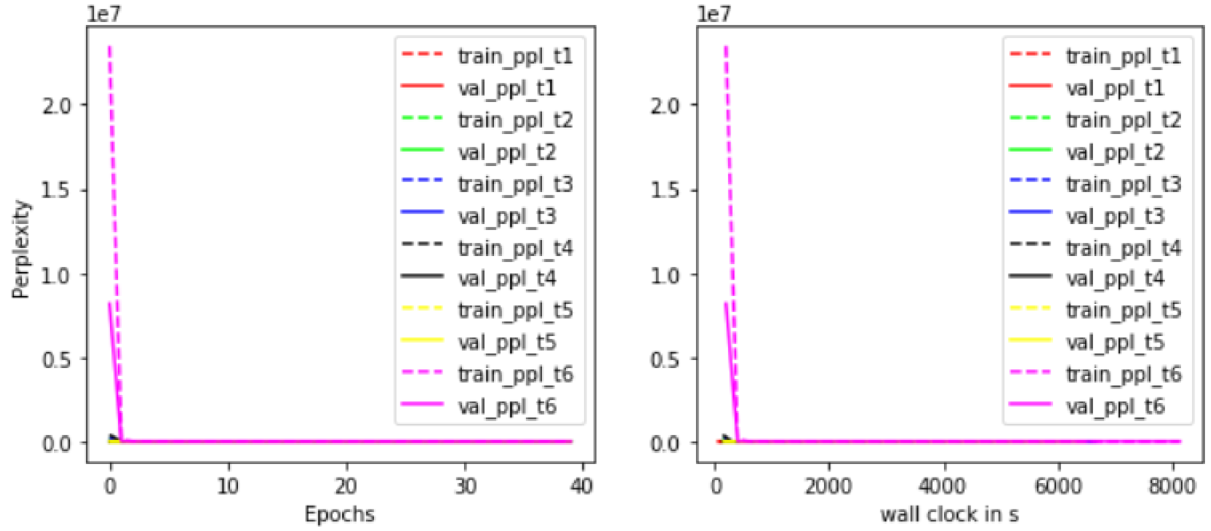


Figure Transformer: Learning curves for the Transformer

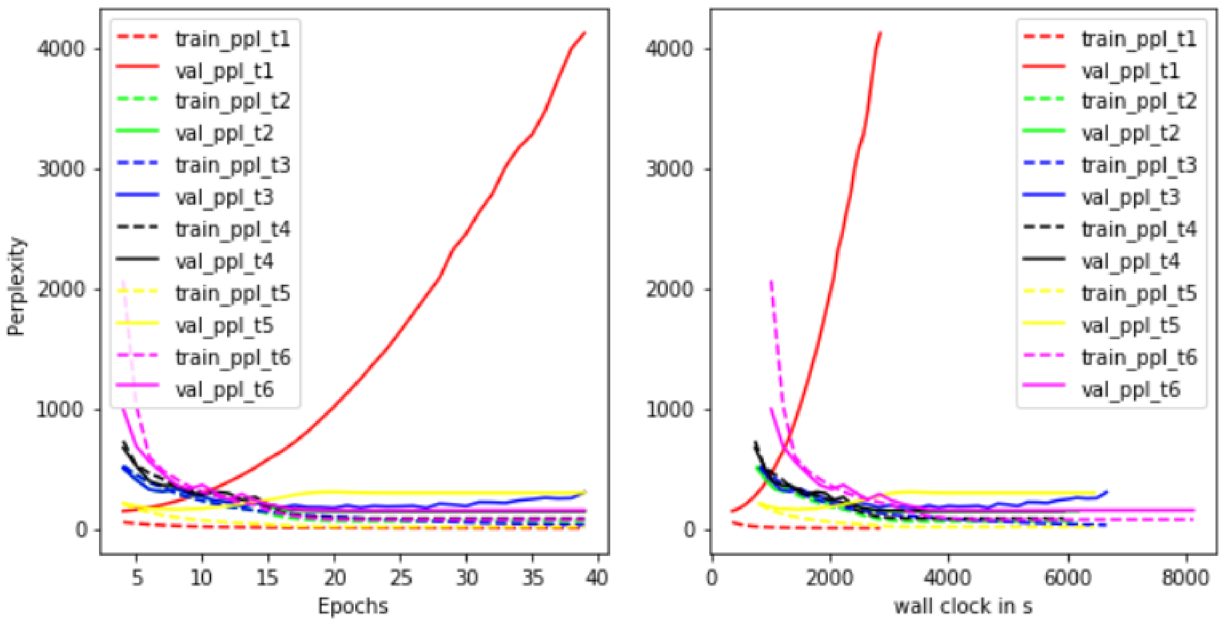


Figure Transformer B: Learning curves for the Transformer after 5 epochs

4.2 Discussion

1. We made several discoveries through these experiments:

   First of all, we were glad to see the **RNN** perform poorly as it is the most basic model. We therefore expected it to yield the worst perplexity and we were proven right as shown in the table.

   Applying the principle made above, we also expected one optimizer to work better than the others. In fact, since the **ADAM** optimizer is the optimizer most commonly used in research papers, we expected it to yield the best results regardless of the architecture. However, we were proven wrong. In particular, when looking at the **GRU** figure, we can see that the **ADAM** optimizer is not always the best. We now realize more clearly that the effectiveness of an optimizer should depend on the architecture as each architecture has a different learning scheme.

We were also surprised to see how effective changing the batch size was with **RNN**. The experiment **r6** shows that increasing the batch size yields better validation perplexity and lower training time. This yields however a higher cost in memory.

2. To analyze the differences between optimizer and architectures, we focus on the figures **RNN B**, **GRU B**, **TRANSFORMER B**, **ADAM B**, **SGD B** and **SCHED B** where we plotted all experiments for each optimizer and architecture. We also focus on the table.

   First of all, the training time doesn't seem to depend on the optimizer. We can see that, for a given set of hyper parameters, the training time is the same across all optimizer.

   We can also notice that the optimizer **ADAM** seems to have the best generalization error in general. According to the table and the plot, it yields the best validation perplexity on average. We believe this to be the main reason why this optimizer is so widely used.

   About synergies, we saw that **RNN** and **SGD** don't go well together (see experiment **r3**). On the other hand, **RNN** work well with **ADAM** optimizer.

3. In terms of training time, the **Transformer** is the fastest model. Therefore, we would choose this model if we were most concerned with wall-clock-time. It has however a bad generalization performance as although the training perplexity is low, the validation perplexity is nearly as high as the **RNN** which is our worst model. As it seems to overfit quite easily, we would be interested in trying this model on more data to see whether in that case the generalization performances would be closer to the training ones or not.

   But with our kind of data set, if we were concerned with the generalization performance instead, we would choose the **GRU** architecture. Indeed, according to the table, they have consistently the lowest validation perplexity across all optimizer and hyper parameters we tried. However, these models have a great computational cost as they are our slowest models to train by far.

   Nevertheless, it doesn't mean that better generalization leads to longer training time in general. Indeed, the **transformer** models have a better generalization performance than the **RNN** models and they are also faster. However, the opposite seems true. Although **transformer** models are fast, the perplexity doesn't change a lot after 20 epochs. The cost of this decent validation perplexity after a few epochs seems to be the incapacity to keep improving significantly through further epochs.

4. The most stable architecture is the **GRU**. According to the table, throughout all our experiments, the validation perplexity remained between 100 and 115. It is thus very stable. Moreover, it remains a decent perplexity as the best perplexity we achieved with other models was over 135.

   On the other hand **RNN** are the least stable. the validation perplexity varies between 149 and 200 with an incredibly high perplexity of nearly 2200 when used with **SGD**.

   The **transformer** models are rather stable, but they performed too poorly with **SGD** optimizer to be considered nearly as stable as **GRU**.

5. We are extremely intrigued about the impact of the number of layers in GRU and RNN models. In particular, we believe that adding layers should not always yield better results. We think that adding too many layers will lead to a vanishing gradient problem, and therefore will lead to worse performances. Thus, we would like to run the GRU models and RNN models with different number of layers, ranging from 2 to 5, in order to analyze the impact of the number of layers. We didn't run these experiments as they would have taken too much time with our computational power. We

actually tried to increase the number of layers, but we had to decrease the hidden size at the same time in order to save computation time, making it hard to really measure the impact of the number of layers.

5 Problem 5

We only had enough time to focus on the question 3, which we found very interesting. First of all, here's our code:

For the generate function in the RNN:

```
def generate(self, input, hidden, generated_seq_len):

    samples = [input]

    for t in range(generated_seq_len):
        x_t = self.embeddings(samples[-1])

        for layer in range(self.num_layers):
            h_t = hidden[layer].clone()
            h_t = torch.tanh(getattr(self, "lin_x_{}".format(layer))(x_t)+
                             getattr(self, "lin_h_{}".format(layer))(h_t))
            x_t = h_t
            hidden[layer] = h_t

        out = self.lin_o(x_t)
        soft = nn.Softmax(-1)
        probas = soft(out)
        samples.append(torch.multinomial(probas, num_samples=1).view(-1))
    return torch.stack(samples)
```

For generate function in the GRU:

```
def generate(self, input, hidden, generated_seq_len):

    samples = [input]

    for t in range(generated_seq_len):

        x_t = self.embeddings(samples[-1])

        for layer in range(self.num_layers):
            h_t = hidden[layer].clone()
            r_t = torch.sigmoid(getattr(self, "lin_rx_{}".format(layer))(x_t) +
                                getattr(self, "lin_rh_{}".format(layer))(h_t))
            z_t = torch.sigmoid(getattr(self, "lin_zx_{}".format(layer))(x_t) +
                                getattr(self, "lin_zh_{}".format(layer))(h_t))
            h_tilde = torch.tanh(getattr(self, "lin_hx_{}".format(layer))(x_t) +
                                 getattr(self, "lin_hh_{}".format(layer))(h_t*r_t))
            h_t = (1-z_t)*h_t + z_t*h_tilde
            x_t = h_t
            hidden[layer] = h_t

        out = self.lin_o(x_t)
        soft = nn.Softmax(-1)
        probas = soft(out)
        samples.append(torch.multinomial(probas, num_samples=1).view(-1))
    return torch.stack(samples)
```

In both cases, we're basically doing a forward pass without dropout. However, where we used to look for the next element in our training sequence, we now take the word that most likely follows the input according to our model.

We then trained a RNN and a GRU with the settings from the question 4.1. At the end of the training, we asked both models to generate a sequence from a input. We decided to use as inputs sequences of 20 '<eos>' tokens. As you can see in our generate function, we need a "*hidden*" parameter. We chose to consider the input batch as is, without any knowledge of any previous sentences, using only the trained weight (ie using the "*init_hidden*" method as our "*hidden*" parameter). We also thought about starting from the hidden states obtained during training. However, we lacked the time to train our models from scratch again. We report some results below. The 40 sentences can be found in the appendix.

Best sequences:

- 2nd sequence from GRU - size 35. Nothing amazing, but still one of the best sentences we found:

  <eos> it should be a new economies <eos> there are a lot of factors in latter says but martin carterpromises to acquire of this year <eos> the concept 's a <unk> <unk> is weak and

- 6th sequence from GRU - size 35. The quantitative facts make almost sense:

  <eos> the main business rate includes <unk> was pushed into normal sales by their N seasons and <unk> <unk> declines to $ N billion from $ N million and department said operating income increased sharply N

- 10th sequence from GRU - size 70. Although some <unk> and <eos> fall in the midst of the second sentence, it makes sense if we ignore them. The first part of the sequence is however utter nonsense:

  <eos> they were reporting that the <unk> collapsed might be hurt by an unwelcome of business-contracts and revenue will be the work position <eos> he added that the ec 's impact of nuclearplants could be brought <eos> by a new technology he headed by earlier this month and the manufacture data running approximately $ N million <eos> holders try to wait with the <unk>sand boosting for more than

Worst sequences:

- 3rd sequence from RNN - size 35. The first part is grammatically wrong and makes no sense:

  <eos> the price groups have is considering what a devaluation of the cloud as not additional spending the u.s. include an wrote again in the world <eos> the measure of the projects insurance whichwould <unk>

- 7th sequence from GRU - size 35. Even the small sequences between the <unk> make no sense:

  <eos> separately 's osaka mercantile trading sold at $ N an ounce up $ N <eos> the los angeles-based <unk> is encouraging <unk> for the audience <eos> none of the <unk> drug containsapproval of arkansas

- 1st sequence from RNN-70. Same as before, even the small chunks make no sense:

  <eos> the two however starts stocks akzo is n't <unk> to i sent to kill the insurance and promptly cut <eos> they freight n't carry in the <unk> of ms. genetic who is <unk> in the physician who

played a hard found the military in other chung <eos> one of the leadership over its legitimacy 'sgame partner on systems is <unk> <unk> on treatment of hand and shippers will be

Interesting sequences:

- 1st sequence from GRU - size 70. It is interesting because we can see that the part of sentences before and after <unk> are completely unrelated:

  <eos> but the most of the same threat is that he receives a major <unk> and interest <eos> iknew it 's not going to realize its staff at the <unk> <eos> when it 's <unk> the kremlin can bringthe upper home to build 's <unk> son i 've gotten in hand or our <unk> to the end of the early1980s when the company announced the lyonnais needed to

- 3rd sequence from GRU - size 70. It is interesting because they talk about "japanese" in two consecuitive sentences, which shows some ability to remember the previous sentences as the models generates the sequence. The first sentence is also very good:

  <eos> we 're expecting a lot of money says jack baldwin the u.s. 's national security adviser<eos> although the array of the japanese like most of its responsibilities has asked commenting on a number of defensive researchers to determine that the company is <unk> for soviet europe <eos> the japanese embassy last week the way was high but the <unk> was replaced <eos> citizens generalmanager and <unk> director of

- 8th sequence from RNN - size 35. It is interesting because it is a good example of the trend both models have. They tend to put the characters N nearby. The first part is also pretty good for an RNN:

  <eos> the financial sector remains on the sweeping growth for example on the interest in the british economy <eos> the tokyo N N crops a week that vigorously gained N bid to N cents a share

As we saw in those examples, usually the best sequences are made up of short sentences. Even among the interesting sequences, the short sentences were usually good. On the other hand, the longer the sentence, the more likely the model is to change subject and make the sequence nonsensical. What we found most interesting are the behaviours with the N and <unk> as pointed out in the interesting examples. We can also see a bias as most sequences are either about politics or economics. This is most likely due to the nature of the training sequences in the penn tree bank. Our models also fail when dealing with named entities (see the 2nd sentence of GRU- size 70). Moreover, both models tend to use the same words a lot (see 6th from GRU - size 70 - repeating of chief executive).

6 Appendix: Generated sentences

6.1 RNN - 10 sequences of size 35

1. <eos> a prices was declining stocks in order for another aggregates for <unk> and bigger suffered those mill bottom of the program 's labor markets <eos> the fed times talking coverage temporarily is all strong a

2. <eos> the <unk> commitment straight to announce the church administration also general some allow of technology game <eos> japan 's export production center matched in series <unk> out the particular in actual economic growth we were

3. <eos> the price groups have is considering what a devaluation of the cloud as not additional spending the u.s. include an wrote again in the world <eos> the measure of the projects insurance which would <unk>

4. <eos> he says the company led that that is enormous holdings in the next contract and he is n't to he 's from their benefits <eos> the officially ford <eos> he has n't be reached for

5. <eos> the floor of am projected prices quickly is inflation down he said in debt and buying by stocks investment interest of in the purchased a wider strategy <eos> the new york N trust N N

6. <eos> the few put the provisions will be a bit a comeback item white house <eos> most of mr. <unk> and is comfortable on good <eos> he was n't father <eos> he also was made both

7. <eos> the company does n't consider n't accurate in some divisions for already except for only to N million shares half of the <unk> buildings used on individual new products <eos> valley one month the network

8. <eos> the financial sector remains on the sweeping growth for example on the interest in the british economy <eos> the tokyo N N crops a week that vigorously gained N bid to N cents a share

9. <eos> any public attorneys but that the <unk> would have become ability to retire <eos> perhaps how legislation <eos> there was the flagship of america started out by about $ N million or between three six

10. <eos> some say the project is so many so <eos> i think <unk> just perhaps that 's <unk> i 'm not going to get its desk <eos> the pie on oct. N editorial he has been

6.2 RNN - 10 sequences of size 70

1. <eos> the two however starts stocks akzo is n't <unk> to i sent to kill the insurance and promptly cut <eos> they freight n't carry in the <unk> of ms. genetic who is <unk> in the physician who played a hard found the military in other chung <eos> one of the leadership over its legitimacy 's game partner on systems is <unk> <unk> on treatment of hand and shippers will be
2. <eos> some companies entered the u.s. that publications played the same system of david <unk> up N cents in N for the wall street journal 's N based in in an interview <eos> in recent years even it said he made he said <eos> meanwhile the offer stemmed from the thomson used also made out of <unk> a but eli lilly co <eos> each of electronics <unk> <unk> <unk> <unk>
3. <eos> one as $ N million or foreign sales and <unk> by <unk> and the new york <eos> and cbs all have <unk> in immediately the state are almost <unk> the spending are <unk> about <unk> to such as much as N years of <eos> on the government will set <unk> <eos> william <unk> the <unk> publisher also picked off loose plan will protect the cycle <eos> if the $ N
4. <eos> the <unk> number of otc market card over the house post <eos> mr. <unk> what he is n't able to <unk> up hearings a soviet leader of them <unk> <eos> the new walker that the time publisher of would be neither is made out of highly former effort to regain state of her brief throwing television in the <unk> <unk> their men <eos> her worry that many <unk> <eos> one
5. <eos> in the latest statement august when the shaky credit talks that public payments too in the full new which each trust N u.s. company has left out of about N issues on taking agencies through the price once a $ form of columbia and <unk> on our credit-card buyer and that mr. stevens said his <unk> needs it holds all even active promise to get a share <eos> what he
6. <eos> the u.s. has underscored on the new cowboys appear because shall generate as fact that things would force be said it would still free telephone <eos> each lobbied to between the future chamber <eos> it says it intends to buy the soviets ' bank <unk> holds most attractive per position <eos> this point just N years in N tons the paper has by a row of eli <unk> telegraph
7. <eos> <unk> mr. freeman was among the fujis could continue distributors to reduce pressure and individual capital markets and petrochemicals affiliates do n't identify any computer completed by matters <eos> thomas find the <unk> general arm of <unk> are used to enter the district market <eos> the new york amount of real estate competition is london N pence N tons in past more sluggish markets <eos> the stocks in yesterday 's
8. <eos> what short rates off some companies in the game is going to be projection that strength <eos> for many in is by the singapore differences have set into survival the possible both that a <unk> calling for the war according to a union discounted is deployed <eos> using the percentage of the day really are so in that organizing <unk> and led since stage columbia 's win N N new
9. <eos> the <unk> $ N million on the bank of another <unk> with them <eos> a daily bureau <eos> it will outcome to receive in N to N in N said he said would scheduled be by a friendly takeover notes <eos> the the company will extraordinary issue to nov. N N in two weeks <eos> <unk> stock <eos> meanwhile accepted this will be would have been <unk> <unk> <eos> meanwhile
10. <eos> the company said its revenue will rise a loss can rise in $ N million <eos> last month new england <unk> N rose N N to N northeast of N two N for the u.s. auto in traffic which was paid a softer analyst at first <unk> said <eos> the <unk> eight a day were reported was weighing N continued growth from N most great building foreign exports and N

6.3 GRU - 10 sequences of size 35

1. <eos> the proposals still is <unk> 's stuff in the u.s. market is a lower investment in american loans which will have surged N N of the new stock market <eos> only that 's N N
2. <eos> it should be a new economies <eos> there are a lot of factors in latter says but martin carter promises to acquire of this year <eos> the concept 's a <unk> <unk> is weak and
3. <eos> mr. thompson replies when the u.s. and ignorance finance group <unk> N to N from its own position <eos> manville launched the president of american consulting co the pse <eos> after the rise in the
4. <eos> i 'm not going here i 'd carry them by the case of <unk> <eos> abbie already <eos> mr. <unk> of his departure is in the way to and his responsibility for clients in fact
5. <eos> the secret of these adjusters has also known <eos> the federation has repeatedly consistently <unk> <eos> that he tells the technology market on the traditional of his <unk> of a decade to expand a site
6. <eos> the main business rate includes <unk> was pushed into normal sales by their N seasons and <unk> <unk> declines to $ N billion from $ N million and department said operating income increased sharply N
7. <eos> separately 's osaka mercantile trading sold at $ N an ounce up $ N <eos> the los angeles-based <unk> is encouraging <unk> for the audience <eos> none of the <unk> drug contains approval of arkansas
8. <eos> but that ended tuesday the <unk> movement in the barber could n't be <unk> affected and the particular 1970s dealers will be keeping utilities from <unk> of baseball hazards around a significant army conference even
9. <eos> if the supreme court will complicate me anyway radical likely should use your military <eos> as the soviets <unk> speaker deukmejian are resisted <unk> for a provision for the purpose of <unk> <eos> junk-bond rewards
10. <eos> the u.s. also cited $ N million in losses it was scrambling to counter the company 's proposed filing against the philadelphia <unk> <eos> it is n't clear that we are looking for ways <eos>

6.4 GRU - 10 sequences of size 70

1. <eos> but the most of the same threat is that he receives a major <unk> and interest <eos> i knew it 's not going to realize its staff at the <unk> <eos> when it 's <unk> the kremlin can bring the upper home to build 's <unk> son i 've gotten in hand or our <unk> to the end of the early 1980s when the company announced the lyonnais needed to

2. <eos> it is a tremendous component of internal delegation in the u.s. the annual credit due dearborn takeovers <eos> in the second term it 's chairman robert j. roberts chief economist and the state the pentagon said <eos> the arm of our government and urban development to lie armstrong as an investment banker as very powerful as a result of wisconsin hub <eos> into the <unk> of the american government the

3. <eos> we 're expecting a lot of money says jack baldwin the u.s. 's national security adviser <eos> although the array of the japanese like most of its responsibilities has asked commenting on a number of defensive researchers to determine that the company is <unk> for soviet europe <eos> the japanese embassy last week the way was high but the <unk> was replaced <eos> citizens general manager and <unk> director of

4. <eos> however a potential is to make a mistake to buy south africa 's new boards without a deal <eos> the stock is so too to indicate that the current slump last week <eos> the delays is still spread by at least it to holds before the price of of assets is n't a decline than any reason until mr. bork would be able to correct <unk> <eos> <unk> the question

5. <eos> the impact is from a <unk> to write allows lesson he said <eos> mr. <unk> has also been served ads making a variety of economics at that are being says <unk> <unk> <unk> assurance for <unk> illustrated new york which becomes a celebration of czechoslovakia 's accessories <eos> john e. <unk> president and chief executive officer of natural gas corp. was elected executive of the merger of penn of the

6. <eos> eugene l. <unk> N years old was named president and chief operating officer was named president and chief operating officer of this systems concern <eos> he succeeds warren a. <unk> who resigned from the board <eos> he succeeds <unk> grand and chief executive officer of aluminum ag and treasurer <eos> harry e. wood N was named chairman of this new president succeeding <eos> mr. <unk> N years old succeeds a

7. <eos> we 're not seeing that we have expecting <unk> said jack <unk> an analyst at schroder de <unk> <eos> these would-be sales have been <unk> <eos> <unk> 's <unk> <eos> <unk> syndicates inc. will add the richer N to N in the first four quarters of the year <eos> pinnacle cfcs thinks the bulk of the most important features of power it will be challenged on <unk> costs in

8. <eos> the collapse consisting and of des a charge is during the past part of the broader market <eos> mr. weil said the bank is found about they who are designed to buy computers as the banking firm drove the securities and some divisions <eos> under the other hand mr. guzman cabrera presented both in january to postpone he bought a host of u.s. trade bank and weisfield 's union <eos>

9. <eos> the letter <unk> a <unk> rumor <unk> partnership and <unk> status water said it holds a N N stake in mortgage-backed securities in the transaction <eos> <eos> n.v the devastating commission said the <unk> reduction is reflects that <unk> through most other companies are died in may to file for bankruptcy <eos> the following the calendar in germans and pricings in san francisco and the two countries were the main

10. <eos> they were reporting that the <unk> collapsed might be hurt by an unwelcome of business contracts and revenue will be the work position <eos> he added that the ec 's impact of nuclear plants

could be brought <eos> by a new technology he headed by earlier this month and the manufacture data running approximately $ N million <eos> holders try to wait with the <unk> sand boosting for more than