From Balustrades to Pierre Vinken: Looking for Syntax in Transformer Self-Attentions

David Mareček and **Rudolf Rosa**
Charles University, Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
Malostranské náměstí 25, 118 00 Prague, Czech Republic
{marecek, rosa}@ufal.mff.cuni.cz

Abstract

We inspect the multi-head self-attention in Transformer NMT encoders for three source languages, looking for patterns that could have a syntactic interpretation. In many of the attention heads, we frequently find sequences of consecutive states attending to the same position, which resemble syntactic phrases. We propose a transparent deterministic method of quantifying the amount of syntactic information present in the self-attentions, based on automatically building and evaluating phrase-structure trees from the phrase-like sequences. We compare the resulting trees to existing constituency treebanks, both manually and by computing precision and recall.

1 Introduction

The classical approach to Natural Language Processing used to be complex pipelines, e.g. (Popel and Žabokrtský, 2010; Manning et al., 2014; Forcada et al., 2011), consisting of multiple steps of linguistically motivated analyses, such as part-of-speech tagging or syntactic parsing, using explicit intermediate representations (e.g. dependency trees) to abstract over the underlying texts.

In recent years, this has changed with the introduction of deep neural end-to-end models, which take raw text as input and produce the desired output directly. Any intermediate representations of the text may emerge during the training of the neural network, and are hidden to us.

We focus on the encoder part of the Transformer architecture (Vaswani et al., 2017), applied to neural machine translation (NMT), as visualizations presented by the authors suggest that its attention heads capture various phenomena such as syntax, semantic roles or anaphora links.

In this work, we analyze the syntactic properties of the self-attention heads both qualitatively and quantitatively. For the quantitative evaluation, we devise a new technique that quantifies the amount of syntactic information by explicitly building constituency trees from the attentions and comparing them with the standard syntactic trees.

Section 3 briefly describes the Transformer encoder architecture and the way we visualize the self-attention matrices using heatmaps. In Section 4, we present our findings from an extensive manual inspection of the heatmaps, identifying several common patterns, including the baluster-like structures which seem to resemble syntactic phrases. To avoid confirmation bias, we proceed by devising a linguistically uninformed tree extraction algorithm (Section 5), which builds a constituency tree based solely on the assumption that the balusters correspond to syntactic phrases. We analyze the resulting parse trees and compare them with standard syntactic trees, both manually and via automatic evaluation. In Section 6, we follow the hypothesis that only some of the attention heads are "syntactic", and try to identify them.

2 Related Work

Initial analyses of syntax captured by neural networks focused on RNNs. Shi et al. (2016) examine how much syntax is learned by RNN encoder by freezing its weights and using a decoder to predict syntactic trees. Adi et al. (2016) examine sentence vector representations by training auxiliary classifiers to take sentence encodings and predict attributes like word order. Linzen et al. (2016) assess the ability of LSTMs to learn syntax by predicting verbal numbers. Blevins et al. (2018) measure the amount of syntax in RNNs by predicting part-of-speech tags and constituent labels.

In the last year, related studies appeared also for the Transformer architecture. Tang et al. (2018) show the Transformer networks perform better

Proceedings of the Second BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP, pages 263–275
Florence, Italy, August 1, 2019.

than RNNs on word sense disambiguation. Zhang and Bowman (2018) show that language models use more syntactic and morphological information than translation models.

Recently, Hewitt and Manning (2019) tried to find syntactic structures in contextual word representations by training simple models on annotated parse trees, concluding that syntactic trees are embedded both in BERT (Devlin et al., 2018) and ELMo (Peters et al., 2018) models. This is also supported by Liu et al. (2019), who successfully trained probes to extract linguistic structures, including syntactic dependencies, from various trained neural networks.

Most existing works train probing models on annotated data (e.g. treebanks). However, such a model may learn to predict the linguistic structure not because it is captured by the network, but because it can be predicted from features preserved from the input, as has been already noted e.g. by Belinkov and Glass (2018). In our work, we try to avoid that risk by not using annotated data for the predictions, but rather looking for structures explicitly present in the network representations.

In a study closely related to ours, Raganato and Tiedemann (2018) also observe syntax-like patterns in Transformer encoder self-attentions, and try to extract syntactic trees without using annotated data (except for taking the root node from the gold annotation). However, they construct dependency trees, while we observe phrase-like rather than dependency-like structures. Moreover, their findings are somewhat inconclusive, as the accuracy of the resulting trees is close to the baseline, while our results are clearly positive. A similar approach was already suggested (but not evaluated) in (Mareček and Rosa, 2018).

3 Transformer NMT Encoder

In the Transformer architecture, Vaswani et al. (2017) came up with several important improvements over the classical attention, including *multi-headed* attention. It features a set of independent attention heads, each deciding on its own to which states to attend. This allows each of the heads to specialize to provide a different type of information or feature (similarly e.g. to CNN filters). The encoder typically uses six multi-head self-attention sub-layers. Each state on a given layer (*output state*) is computed from a concatenation of the result of applying a set of attention heads to the states on the previous layer (*input states*), passed through a feed-forward layer. This may allow the encoder to do more advanced multi-step processing, such as aggregating the information about several subwords into one position and then attending to this position on the higher layers.

Another notable feature of the Transformer encoder is the use of residual connections, which transport the source subword embeddings forward, bypassing the self-attention mechanism, and get averaged with the outputs of the self-attention. This ensures that the *output state* at each position retains a significant amount of the corresponding source subword embedding, supporting the usual shortcut of assuming that the hidden states can be thought of as representations of the underlying subwords (in the context of the sentence).

3.1 Encoder Self-Attention Visualization

We focus on exploring multi-head self-attentions of the encoder. We use a natural visualization of self-attention heads using square matrix heatmaps (Figure 1), going from black (attention weight = 0) to white (attention weight = 1). The subwords that correspond to the rows and columns are printed alongside the matrix. The rows correspond to *output states*, and the columns to *input states*; as the *output states* attend to *input states*, the softmaxed attention weights on each row sum to 1.

Note that the visualizations may be deceiving in several aspects. It is important to understand that the fact that a given head at a given position on a given layer attends to a position of a specific subword does *not* mean that the resulting hidden state will simply contain the representation of that subword, for several reasons:

- The input to the self attention is the output of the previous layer, i.e. a hidden state, presumably but not necessarily representing the subword at this position to some extent, and usually mixing in information about other subwords in the sentence.
- The hidden states emitted from each layer are the outputs of a feed forward network that takes a concatenation of outputs from all of the heads on that layer as input, and can thus mix them, ignore them, only use parts of them, etc.

3.2 Experiment Setup

We analyze the Transformer NMT encoders for the following three languages: English (en),

en-de	33.5	en-fr	45.2	fr-de	24.3
de-en	39.8	fr-en	42.1	de-fr	32.9

Table 1: BLEU scores measured on the test data.

French (fr), and German (de). We selected those particular languages because they are available in the Europarl corpus[1] (Koehn, 2005) comprising large high-quality multiparallel data, and because constituency syntax parse trees can be obtained for them by the Stanford parser (Klein and Manning, 2003) out-of-the-box.[2]

As we want to explore a state-of-the-art setup, we use the Transformer model (Vaswani et al., 2017) as reimplemented by Helcl et al. (2018) in the Neural Monkey framework[3] in standard setting: 6 encoder and decoder layers, 16 attention heads, embedding size of 512, hidden-layers' size of 4096, dropout 0.9, and batch size 30.

We train the translator for all 6 source-target language pairs (en-fr, en-de, fr-en, fr-de, de-en, de-fr).[4] From the Europarl corpus, we take first 1,000 sentences as development data, last 1,000 sentences as evaluation data, and the remaining 486,272 sentences for training. Table 1 lists the BLEU scores of the systems. All inspections and evaluations, both manual and automatic, have been performed on the evaluation data.

The data are tokenized by the Stanford Tokenizer[5] to make the tokens consistent with the constituency trees with which we will compare our results. We then build a shared dictionary of 100,000 BPE subword units (Sennrich et al., 2016) on the concatenated training data of all three languages, append an EOS symbol to each sentence, and train the translation model.

4 Manual Analysis of Attention Matrices

On a small sample of 10 sentences and for each language pair, we created the heatmaps for all 16 attention heads of all 6 encoder layers. Six heatmaps for one sentence from the en→de encoder are shown in Figure 1; all 96 of them are enclosed in the Appendix.

A general observation is that the attentions are nearly always very peaked. Even though the attention mechanism was designed as soft, most attention heads concentrate nearly all of the attention at each *output state* onto just one *input state*.

In the following subsections, we list all of the distinctive patterns that we have identified.[6] An important thing to note is that typically, a head behaves consistently across all sentences, i.e., for a given head on a given layer of a given trained Transformer encoder, we typically see the same attention patterns across all sentences.

4.1 Diagonals

Especially at the first encoder layer, there often appear various simple *diagonal* heads.

Typically, each *output state* attends to the *input state* at the same position. This may serve to pass the subword information to the higher layers.

In some cases, most of the *output states* attend to the corresponding *input states*, but some of them attend elsewhere. The role of such *partial diagonal* may be looking for a specific phenomenon that only occurs for some of the *output states*.

Often, individual *output states* attend to preceding or following *input states*, forming a *parallel diagonal* (Figure 1b). Sometimes the heads attend further, e.g. to the "pre-previous" *input state*.

4.2 Balustrades

The most frequent pattern, appearing in about 2/3 of the attention heads, are *balustrades* – a series of vertical bars, typically placed at the diagonal, which resemble the balusters of a staircase railing. Examples of such balustrades are shown in Figure 1c,d,e. The balustrades are often placed upwards or downwards from the main diagonal.

We observe that different heads contain balusters of different lengths. For longer balusters, the *input state* that they attend to often corresponds to a punctuation or a conjunction; often there are also heads that attend exclusively or almost exclusively to the sentence-final punctuation.

We have noticed that in many cases, the sequence of subwords spanned by a baluster may be understood as a syntactic phrase (e.g. a noun and its determiner, or a syntactic clause between

[1] http://data.statmt.org/wmt18/translation-task/training-parallel-ep-v8.tgz

[2] https://nlp.stanford.edu/software/lex-parser.html

[3] https://github.com/ufal/neuralmonkey

[4] We intersect the English-German and English-French parallel corpora using English as pivoting language.

[5] https://nlp.stanford.edu/software/tokenizer.shtml

[6] We observe all patterns which Raganato and Tiedemann (2018) identified, i.e. diagonals and attending to the end of the sentence, but also other patterns which they did not observe.

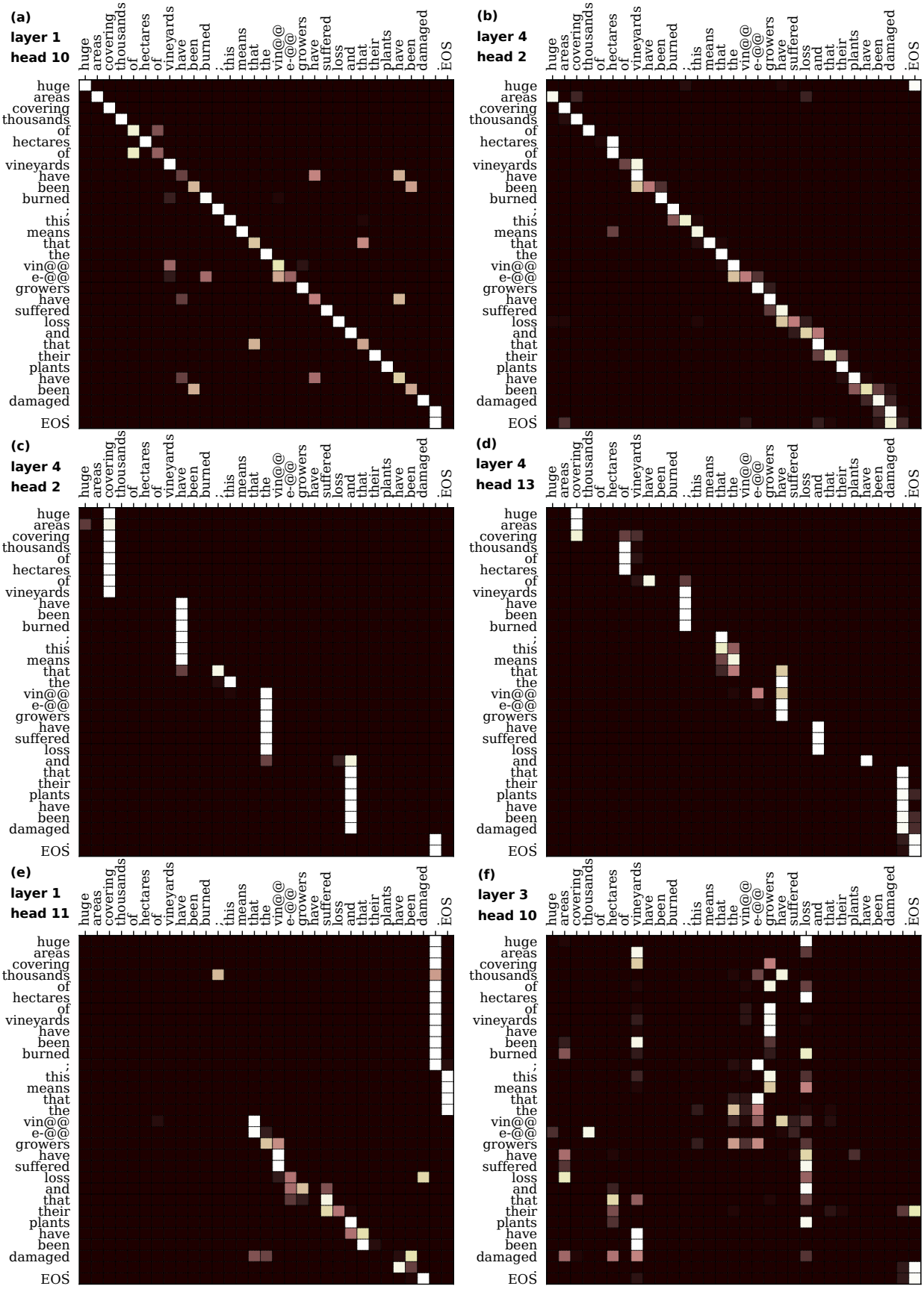

Figure 1: Heatmaps of selected attention heads showing different patterns. There are *diagonal* patterns in (a) and (b), *balustrades* in (c) and (d), a combination in (e), and rather scattered attention in (f).

two commas). Furthermore, by looking at multiple attention heads at once, we can interpret the balusters of various lengths spanning the same subwords as shorter phrases nested within longer phrases. This leads us to the idea of constructing a constituency tree from the nested phrases, and comparing it with classical syntactic constituency trees (see Section 5).

4.3 Equal or Similar Subwords

There is typically one or two heads where each *output state* attends to all instances of the same subword, usually with a more or less uniform distribution (see the subwords "of", "have" and "that" in Figure 1a). We have also seen these heads to sometimes attend to very similar but not identical subwords (e.g. singular and plural).

4.4 The Rest

Admittedly, for about 1/5 of the attention heads, we have not identified any clear pattern, and thus have no hypothesis as for the function of such heads. Sometimes, the head shows some of the behaviours only for some of the *output states*; sometimes we do not see even such partial patterns (Figure 1f).

5 Extracting Constituency Trees

Our aim is to analyze whether syntactic structures seem to be captured by Transformer self-attentions, to what extent, and of what kind. As explained in the previous section, we often observe balusters of various lengths in the attention heatmaps, which can be interpreted as nested syntactic phrases. In this section, we try to measure to which extent this interpretation seems to be valid.

For this purpose, we devise a linguistically uninformed transparent deterministic algorithm to extract binary constituency trees from the balusters (Section 5.1). We automatically evaluate the results by comparing them with classical syntactic trees, generated by a standard syntactic parser (Section 5.2), to see whether the observed structures seem to capture syntax as we know it. We discuss the results in Section 5.3.

5.1 Tree Extraction Algorithm

We now explain how we construct constituency trees from the balusters in the attention matrices.

Our goal is not to optimize our algorithm towards producing good syntactic trees. Rather, we try to keep our algorithm linguistically uninformed, to reveal only what really is captured by the self-attentions. Therefore, we:

- build binary constituency trees, as this is quite a basic way to represent nested phrases,
- use information from all attention heads, not only those which seem to capture syntax,
- keep the number of other hyperparameters minimal and set them to the most uninformed values, rather than tuning them,
- do not train or tune the tree extraction in any way (unlike most related work).

The first step is to identify the balusters. We have previously described a baluster as a sequence of *output states* attending to a single *input state*. The attentions are typically very peaked, with nearly all of the attention mass concentrated onto one *input state*. However, as the attentions are soft, each of the *output states* in fact attends to all of the *input states* to some extent. We thus "harden" the soft attention matrix A' by only keeping the maximal attention weight on each row of the attention matrix, setting all the other weights to 0:

$$A_{o,i} = \begin{cases} A'_{o,i} & \text{if } A'_{o,i} = \max_{j \in [1,N]} A'_{o,j} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where i is the *input state* index, o is the *output state* index, and N is the sentence length.

Next, we extract candidate phrases from the balusters and weight them. From each baluster, we extract only the candidate phrase corresponding to the full length of the baluster. The weight of the phrase corresponds to the average attention that *output states* in the phrase give to the common *input state* they attend to (i.e. the average brightness of the points in the baluster). If the same phrase appears in multiple attention matrices, their scores are summed together. The weight of the phrase spanning the a-th to b-th subwords thus is:

$$w'_{a,b} = \sum_{h \in H_{a,b}} \frac{\sum_{o \in [a,b]} A^h_{o,i_h}}{b - a + 1} \quad (2)$$

where $H_{a,b}$ is the set of attention heads containing a baluster spanning the *output states* a to b, A^h is the hardened attention matrix for head h, and i_h is the *input state* attended by the baluster in head h.

The weights defined in this way are unbalanced, giving more importance to shorter phrases, as they are more frequent in the attention matrices.

We thus equalize the weights so that the average weight of all phrases of the same length equals 1:

$$w_{a,b} = \frac{w'_{a,b} \cdot |P^{b-a+1}|}{\sum_{(c,d) \in P^{b-a+1}} w'_{c,d}} \quad (3)$$

where P^k is the index pair set of all extracted phrases of length k.

To construct the constituency tree from the phrases, we use the CKY dynamic programming algorithm (Ney, 1991), which searches for the highest scoring constituency tree in $O(n^3)$.

For each tree spanning the a-th to b-th subword, we define its score $s_{a,b}$ recursively by finding a separator k, $a \leq k < b$, that maximizes the average of scores and weights of the two subtrees with spans (a, k) and $(k+1, b)$:

$$s_{a,b} = \max_k \frac{s_{a,k} + s_{k+1,b} + w_{a,k} + w_{k+1,b}}{4}. \quad (4)$$

The initial scores for single-subword subtrees are set to 1. The averaging then keeps the scores equalized – subtrees then have the same power regardless of the size of their spans.

The CKY algorithm works bottom up, starting with the trivial single-subword trees, and then iteratively computing the values of larger subtrees based on the values precomputed in previous steps. Together with the score of each tree, the algorithm also stores the k from Equation 4, which defines the highest scoring pair of subtrees covering the same span. Once the algorithm reaches the tree covering the whole sentence, it recursively returns the highest scoring tree based on the stored values of the highest scoring subtrees.

5.2 Automatic Evaluation

To evaluate the syntacticity of the Transformer self-attentive encoder, we extract the constituency trees using our tree extraction algorithm for the 1,000 sentences of our evaluation set; we will refer to these as *extracted trees*.

We then induce syntactic trees for these sentences with the Stanford Parser. We use the factored lexicalized parsing models distributed together with the parser, which had been trained on standard constituency treebanks of the languages – English Penn Treebank (Marcus et al., 1993), German Negra Corpus (Skut et al., 1999), and French Treebank (Abeillé et al., 2003). We post-process the trees in the following way:

1. remove phrase labels
2. wrap each word into a single-word phrase
3. split words into subwords
4. flatten phrases containing only one immediate subphrase or only one subword

We show an example of applying this procedure:

0. (S (VP vinegrowers suffer))
1. ((vinegrowers suffer))
2. (((vinegrowers) (suffer)))
3. (((vin- e- growers) (suffer)))
4. ((vin- e- growers) suffer)

We will refer to the resulting trees as *parse trees*.

We compare the extracted trees with the parse trees, assuming that the more similar they are, the more syntactic the Transformer encoder is.

We calculate the *precision* of the extracted tree as the proportion of its phrases that are "correct" in the sense that they are consistent with the parse tree, not crossing any of its phrases. (For the sake of this analysis, we only consider one possible way of capturing syntax, as defined in the respective treebanks; we discuss that in Section 5.3.)

Let P be the parse tree, an extracted phrase e is correct if and only if:

$$\forall p \in P : (p \cap e = \emptyset) \vee (p \subseteq e) \vee (e \subseteq p). \quad (5)$$

Recall is computed inversely, as the proportion of phrases in the parse tree that are consistent with the extracted tree. We compute the total precision and recall as an average over all extracted phrases in all the trees, and also report their harmonic mean (*F1*).

The results of the evaluations for all three source languages are shown in Table 2. To put them into perspective, we also report scores for several uninformed parsing baselines:

1. *rbal*: balanced binary tree aligned right
2. *lbal*: balanced binary tree aligned left
3. *rand.init*: our proposed algorithm using randomly initialized Transformer weights

Examples of the *lbal* and *rbal* baselines are shown in Figure 2.

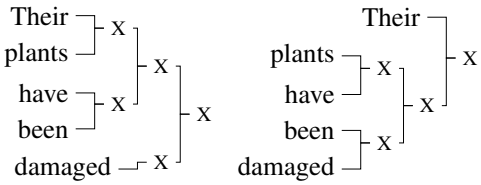

Figure 2: Left (*lbal*) and right (*rbal*) balanced binary tree baselines.

English

system	precision	recall	F1 score
rbal	30.1%	24.3%	26.8%
lbal	27.8%	20.8%	23.8%
rand.init	25.1%	20.0%	22.3%
en → de	35.4%	30.6%	32.8%
en → fr	35.4%	30.2%	32.6%

German

system	precision	recall	F1 score
rbal	39.1%	31.3%	34.8%
lbal	38.1%	27.6%	32.0%
rand.init	33.7%	25.9%	29.3%
de → en	46.1%	39.6%	42.6%
de → fr	46.7%	40.9%	43.6%

French

system	precision	recall	F1 score
rbal	34.3%	28.7%	31.3%
lbal	32.5%	25.4%	28.5%
rand.init	26.1%	24.4%	25.3%
fr → en	44.4%	39.7%	41.9%
fr → de	46.9%	41.7%	44.2%

Table 2: Scores of baseline trees and our extracted trees using all attention heads, evaluated against standard syntactic parse trees.

5.3 Discussion of Results

The F1 scores of the trees extracted from the attention matrices are 6 to 13 percentage points higher than the best baselines, showing that some syntax is indeed captured by the Transformer encoder.

For English, the scores are notably lower than for the other languages. Manual inspection has shown that this is mostly due to the English parse trees being strongly right-branching, while the other treebanks use flatter, more balanced trees, mainly due to different annotation styles of the treebanks. The trees extracted from the attention matrices are similar for all of the languages, and resemble the German or French parse trees more than the English ones. However, a part of the score differences may also be due to a differing syntacticity of the individual encoders, as can be seen from the differing scores for fr→en and fr→de.

Figure 3 shows an example of a tree extracted from the en→de encoder (the sentence is the same as in Figure 1). We can see that many of the subtrees seem to make sense syntactically, both smaller ones, such as "[have been] damaged", as well as larger ones, such as the tree spanning "huge... vineyards". Some are questionable, but not necessarily wrong, e.g. "[the vine-] growers".

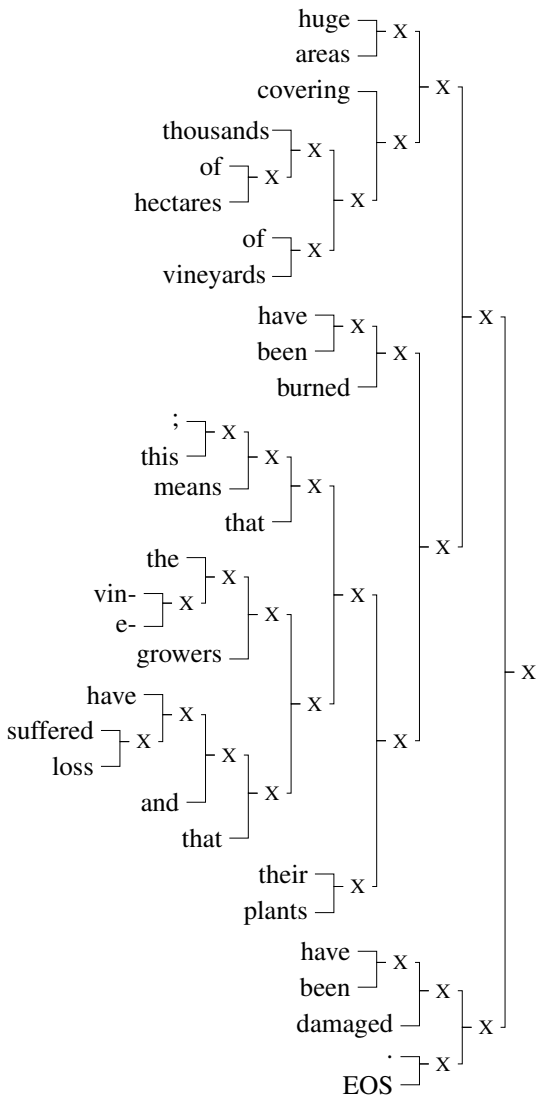

Figure 3: A constituency tree generated by our tree extraction algorithm from the attention matrices of the en-de encoder for the 4th sentence of the evaluation set.

A clear limitation of our automatic evaluation method is that it only evaluates whether the structures match those of the syntactic formalism of the standard treebank, but it cannot appreciate alternative structures that also make sense syntactically. However, this issue is hard to solve without a significant amount of manual work.

Nevertheless, some structures clearly do not correspond to the syntactic structure of the sentence, regardless of the syntactic formalism that we adhere to. E.g. the phrases "their plants" and "have been damaged" belong together, but they are separated in the extracted tree all the way to the root. The reason we find these incorrect structures in the extracted trees may be that we are using all

the encoder attention matrices in the extraction algorithm, even though not all of the attention heads seem to behave syntactically; we investigate this to some extent in the next section. However, it is also quite likely that the encoder only captures some parts of the syntactic structure of the sentence, not a full syntactic tree – especially given the fact that the model is trained to do machine translation, and may thus have no reason to capture structures irrelevant for this task. Moreover, classical syntactic trees are by far not the only possible way of capturing syntax, and it is quite likely that the syntax captured by the self-attentive encoder should be understood differently.[7]

6 Selecting Syntactic Heads

As we have discussed in Section 4, there is a range of different types of attention heads. In our interpretation, some of them, especially the *balustrades*, seem to capture syntactic structures, while others seem not to do so. A logical step thus is to try to identify the syntactic heads, and only use those for the tree extraction.[8]

We propose to use the automatic evaluation as the criterion for selecting the "syntactic" heads. We suggest two greedy approaches: *head addition*, and *head ablation*.

In the *head addition* approach, we start with an empty set of heads and then iteratively add the heads one by one, maximizing the precision of the extracted trees in each step, until we have the set of all heads. We then identify the highest scoring head combination that we encountered.

The *head ablation* approach is the logical inverse; we start with all the heads and iteratively remove them until we end up with only one head.

We ran the selection algorithms using only the first 100 sentences. The setups selected as best by the algorithm were then evaluated on the full evaluation set. As the *head addition* consistently outperformed *head ablation* by approximately 2 percentage points, we only report the evaluation of the *head addition* in Table 3.

	improvement in		
system	precision	recall	F1 score
en → de	+9.48%	+7.01%	+8.10%
en → fr	+8.43%	+6.23%	+7.19%
de → en	+4.60%	+2.06%	+3.13%
de → fr	+5.96%	+1.76%	+3.52%
fr → en	+11.58%	+8.54%	+9.91%
fr → de	+12.16%	+8.63%	+10.20%

Table 3: Evaluation of syntactic heads subselection. Score gains over the base tree extraction as reported in Table 2, in percentage points.

L	1	2	3	4	5	6
P	36%	3%	10%	10%	19%	21%

Table 4: Average proportion of attention head layers in the best subselection setups for all language pairs. L is the number of the layer, P is the proportion of the selected heads that come from the given layer.

We can see improvements in F1 ranging from 3 to 10 percentage points, showing that better syntactic trees can be extracted by subselecting the heads. However, we are perhaps overtuning the setup, and the reported numbers are thus probably somewhat inflated. Therefore, we are reluctant to draw any strong conclusions from the results.

Nevertheless, the meta-analysis of the heads selected as syntactic is of interest. For each of the language pairs, between 18 and 32 heads of the total 96 were selected. However, these are not evenly distributed across the layers. As we show in Table 4, on average, one third of the selected heads come from the first layer, which mostly contains diagonals and short balusters; the last two layers, which contain a lot of balusters of varied lengths, each contributes one fifth of the heads.

7 Conclusion

We analyzed the Transformer encoder self-attention, identifying baluster structures resembling syntactic phrases. We devised a transparent linguistically uninformed algorithm for extracting constituency trees from the balusters, compared the resulting trees with standard syntactic parse trees, and showed that syntax is indeed captured.

Acknowledgments

This work has been supported by the grant 18-02196S of the Czech Science Foundation.

[7] For example, the syntactic structure could be quite flat, with shorter phrases or treelets joined into a linked list, rather than a complex tree structure with long-distance relations. Also, we have noted that connectors, such as punctuation and conjunctions, often seem to be part of both of their neighbouring phrases, which could lead to a formalism using partially overlapping phrases. We intend to investigate this in future.

[8] However, once we start subselecting only some of the heads, we are clearly introducing our expectations about the syntactic structures to be found into the process – we are now contaminating the so far linguistically uninformed approach with our notion of "good" or "syntactic" phrases.

References

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a treebank for french. In *Treebanks*, pages 165–187. Springer.

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2016. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *CoRR*, abs/1608.04207.

Yonatan Belinkov and James Glass. 2018. Analysis methods in neural language processing: A survey. *CoRR*, abs/1812.08951.

Terra Blevins, Omer Levy, and Luke Zettlemoyer. 2018. Deep RNNs encode soft hierarchical syntax. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 14–19, Melbourne, Australia. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Mikel L. Forcada, Mireia Ginestí-Rosell, Jacob Nordfalk, Jim O'Regan, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Felipe Sánchez-Martínez, Gema Ramírez-Sánchez, and Francis M. Tyers. 2011. Apertium: a free/open-source platform for rule-based machine translation. *Machine Translation*, 25(2):127–144.

Jindřich Helcl, Jindřich Libovický, Tom Kocmi, Tomáš Musil, Ondřej Cífka, Dušan Variš, and Ondřej Bojar. 2018. Neural monkey: The current state and beyond. In *The 13th Conference of The Association for Machine Translation in the Americas, Vol. 1: MT Researchers' Track*, pages 168–176, Stroudsburg, PA, USA. The Association for Machine Translation in the Americas, The Association for Machine Translation in the Americas.

John Hewitt and Christopher D. Manning. 2019. Structural Probe for Finding Syntax in Word Representations. In *Proceedings of NAACL 2019*.

Dan Klein and Christopher D Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in neural information processing systems*, pages 3–10.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand. AAMT, AAMT.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. *CoRR*, abs/1903.08855.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

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

David Mareček and Rudolf Rosa. 2018. Extracting syntactic trees from transformer encoder self-attentions. In *Proceedings of the First Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 347–349, Stroudsburg, PA, USA. The Assotiation of Computational Linguistics.

Hermann Ney. 1991. Dynamic programming parsing for context-free grammars in continuous speech recognition. *Trans. Sig. Proc.*, 39(2):336–340.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Martin Popel and Zdeněk Žabokrtský. 2010. TectoMT: Modular NLP framework. In *Lecture Notes in Artificial Intelligence, Proceedings of the 7th International Conference on Advances in Natural Language Processing (IceTAL 2010)*, volume 6233 of *Lecture Notes in Computer Science*, pages 293–304, Berlin / Heidelberg. Iceland Centre for Language Technology (ICLT), Springer.

Alessandro Raganato and Jörg Tiedemann. 2018. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 287–297. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725. Association for Computational Linguistics.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *EMNLP*, pages 1526–1534.

Wojciech Skut, Hans Uszkoreit, and Thorsten Brants. 1999. Syntactic annotation of a german newspaper corpus. In *ATALA sur le Corpus Annotés pour la Syntaxe Treebanks, June 18-19*, pages 69–76, Paris, France. o.A.

Gongbo Tang, Mathias Müller, Annette Rios, and Rico Sennrich. 2018. Why self-attention? a targeted evaluation of neural machine translation architectures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4263–4272. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6000–6010. Curran Associates, Inc.

Kelly W. Zhang and Samuel R. Bowman. 2018. Language modeling teaches you more syntax than translation does: Lessons learned through auxiliary task analysis. *CoRR*, abs/1809.10040.

Appendix: Visualization of all attention heads

We provide visualisations of encoder's self-attention heads for English source sentence *"Huge areas covering thousands of hectares of vineyards have been burned; this means that the vin@@ e-@@ growers have suffered loss and that their plants have been damaged."*, when translating into German.

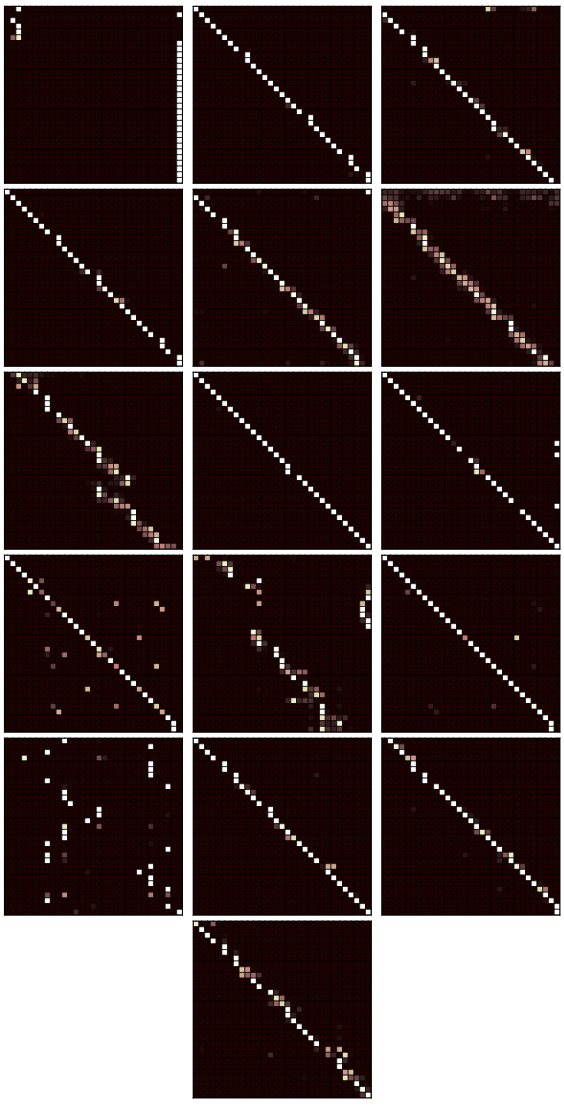

Figure 4: Layer 1

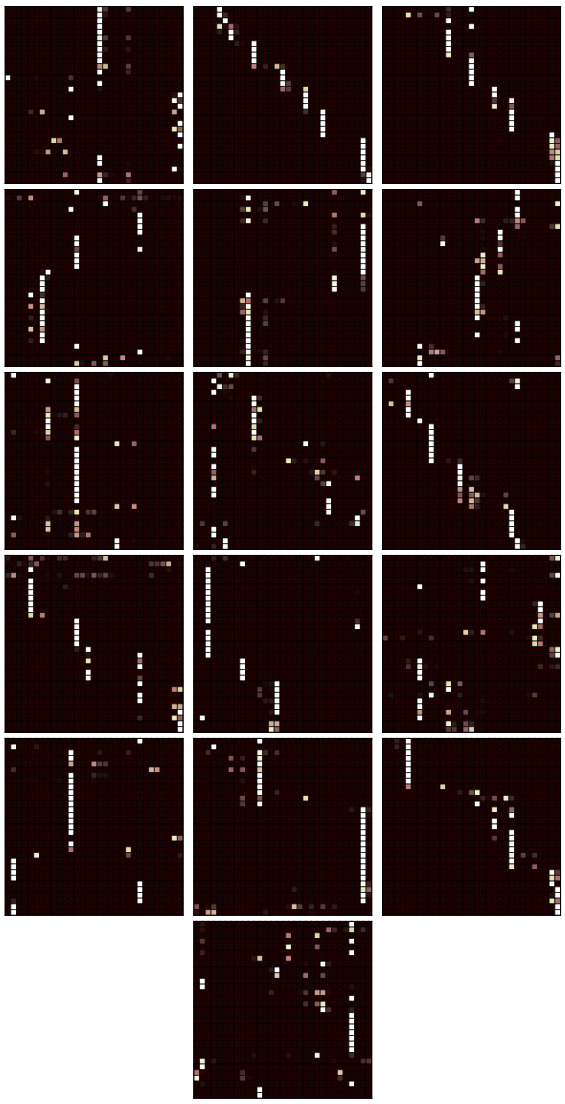

Figure 5: Layer 2

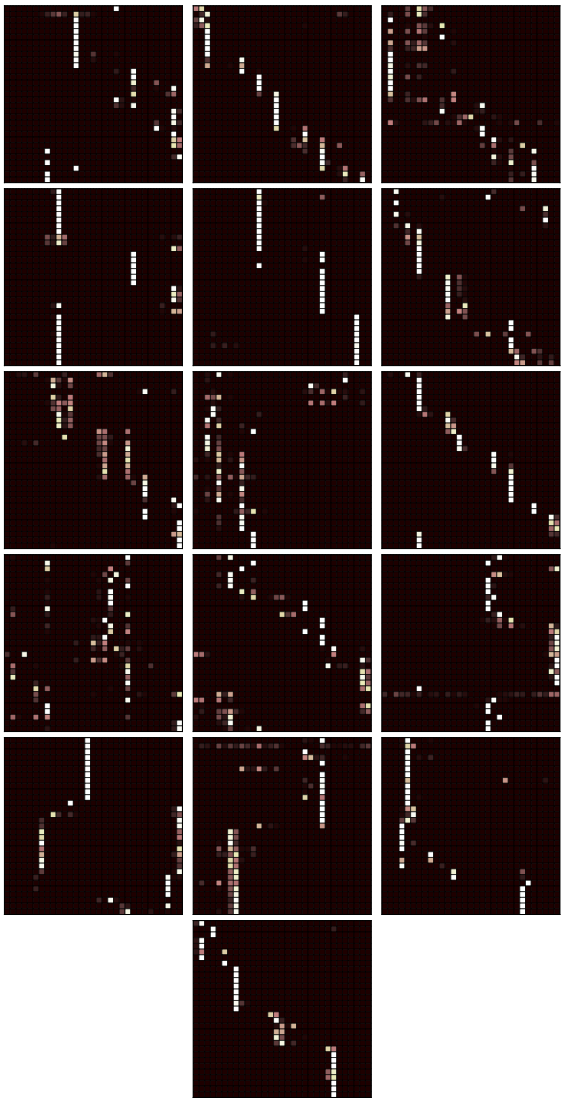

Figure 6: Layer 3

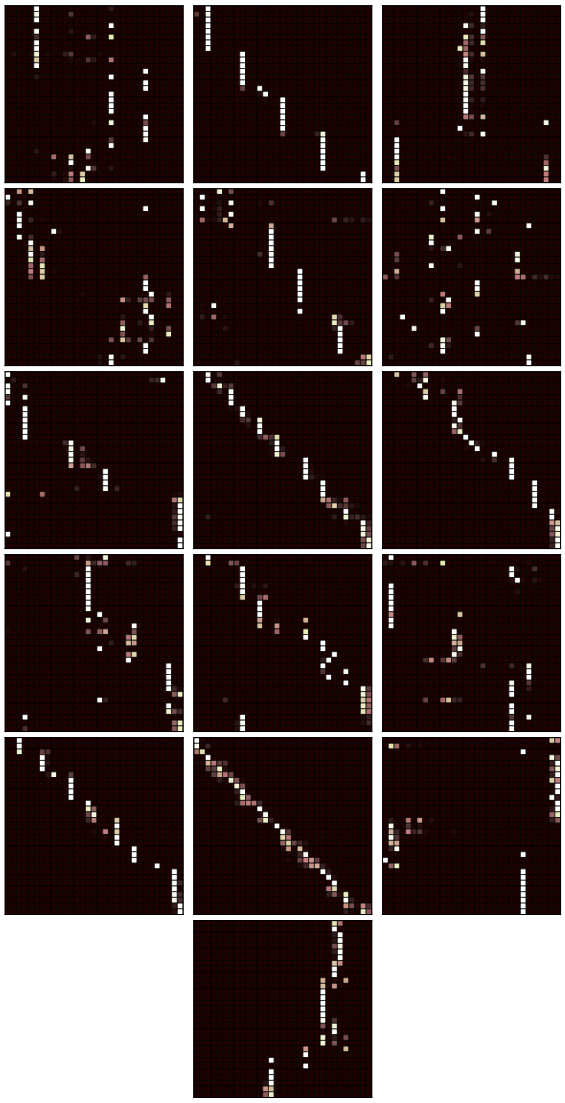

Figure 7: Layer 4

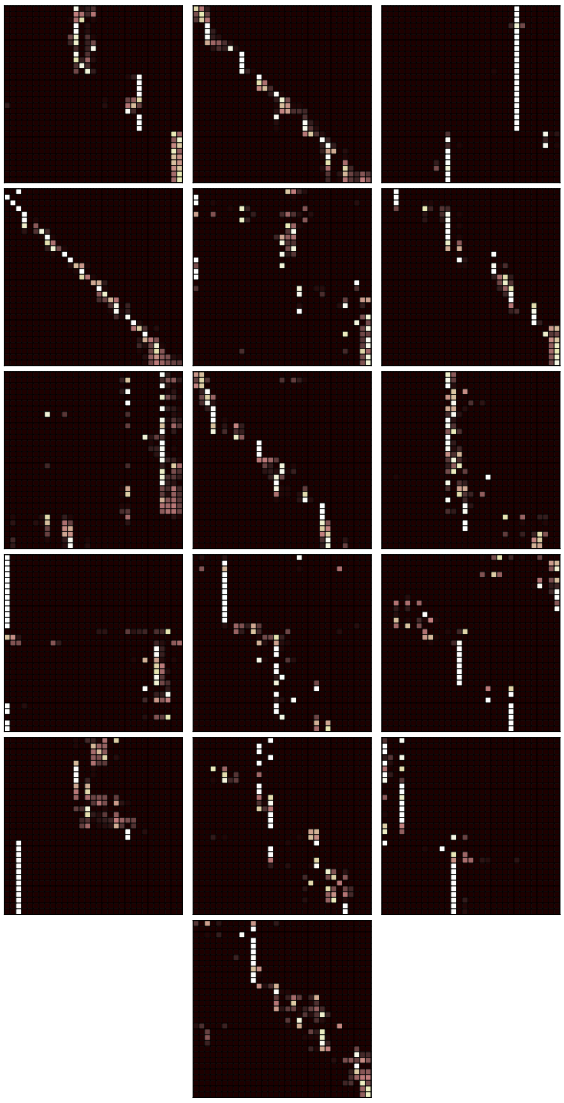

Figure 8: Layer 5

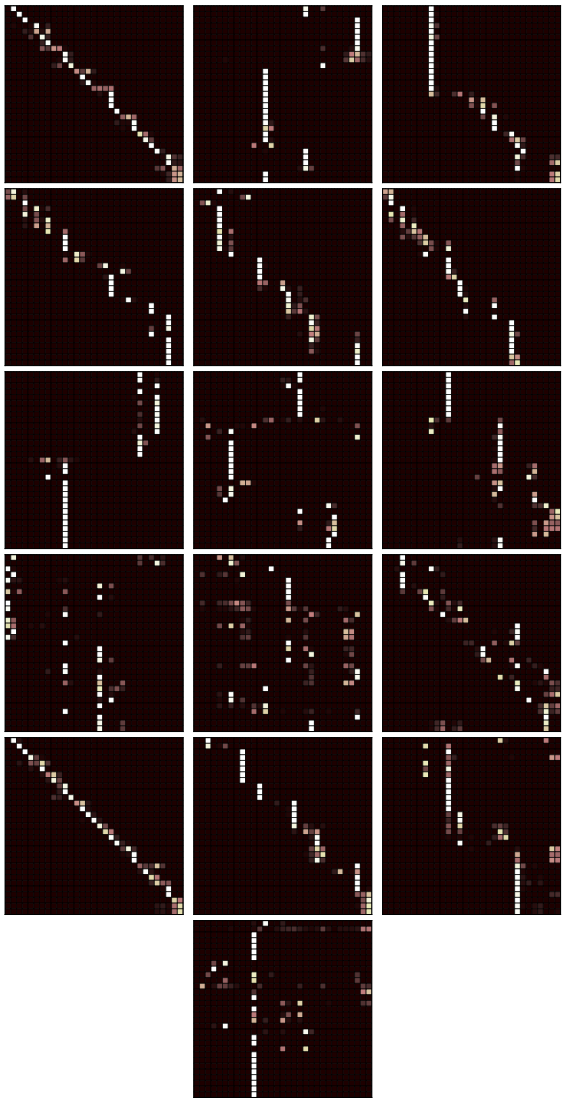

Figure 9: Layer 6