6 APPENDIX A. TRAINING PROTOCOLS AND HYPERPARAMETER CHOICES.

The fruit fly network was trained on the OpenWebText Corpus (Gokaslan & Cohen, 2019), which is a 32GB corpus of unstructured text containing approximately 6B tokens. Individual documents were concatenated and split into sentences. A collection of w-grams were extracted from each sentence by sliding a window of size w along each sentence from the beginning to the end. Sentences shorter than w were removed. The vocabulary was composed of $N_{\text{voc}} = 20000$ most frequent tokens in the corpus.

Training was done for N_{epoch}. At each epoch all the w-grams were shuffled, organized in mini-batches, and presented to the learning algorithm. The learning rate was linearly annealed starting from the maximal value ε_0 at the first epoch to nearly zero at the last epoch.

The training algorithm has the following hyperparameters: size of the KC layer K, window w, overall number of training epochs N_{epoch}, initial learning rate ε_0, minibatch size, and hash length k. All models presented in this paper were trained for $N_{\text{epoch}} = 15$. The optimal ranges of the hyperparameters are: learning rate is $\varepsilon_0 \approx 10^{-4} - 5 \cdot 10^{-4}$; $K \approx 200 - 600$; $w \approx 9 - 15$; minibatch size $\approx 2000 - 15000$; hash length k is reported for each individual experiment.

7 APPENDIX B. COMPARISON WITH BINARIZED GLOVE AND WORD2VEC.

Method	Hash Length (k)						Hash Length (k)					
	4	8	16	32	64	128	4	8	16	32	64	128
	MEN (69.5/68.1)						**WS353** (64.0/47.7)					
Ours	**34.0**	**49.9**	**55.9**	56.7	55.3	51.3	**43.2**	**52.1**	**55.3**	**57.4**	**60.3**	51.7
LSH	16.9	23.7	35.6	42.6	53.6	63.4	8.2	20.7	30.0	34.7	43.9	50.3
RandExp	27.5	37.7	46.6	**57.6**	**67.3**	**71.6**	20.9	32.9	41.9	48.4	57.6	61.7
ITQ	0.1	7.7	10.5	16.5	30.4	50.5	-6.6	-6.1	-2.4	-4.4	6.1	24.8
SH	9.4	17.0	22.9	37.6	52.9	65.4	15.4	14.1	19.5	32.3	43.1	58.4
PCAH	12.5	21.8	27.6	39.6	53.4	68.1	6.4	6.3	20.6	33.9	49.8	**62.6**
NLB	-	-	-	-	46.1	63.3	-	-	-	-	30.1	44.9
	SIMLEX (31.5/29.8)						**RW** (46.8/31.4)					
Ours	**13.4**	16.5	**22.8**	22.1	21.1	17.0	11.0	**22.6**	25.8	36.9	38.6	35.2
LSH	6.8	11.9	17.0	21.2	26.8	30.9	10.8	16.3	21.8	27.8	36.3	45.0
RandExp	10.4	**17.2**	**22.8**	**28.5**	**32.4**	**35.2**	19.9	21.3	**30.9**	**40.5**	**47.6**	**53.3**
ITQ	7.0	1.6	4.3	5.5	11.8	18.2	13.7	5.3	6.6	6.9	12.5	26.5
SH	9.3	15.6	15.9	17.0	23.1	31.2	**22.6**	21.5	24.3	28.8	36.1	45.8
PCAH	4.4	10.3	11.0	17.3	24.1	31.6	12.4	16.7	21.5	30.3	36.9	44.4
NLB	-	-	-	-	20.5	31.4	-	-	-	-	25.1	34.3
	RG (74.2/67.6)						**Mturk** (57.5/61.9)					
Ours	24.0	40.4	**51.3**	62.3	63.2	55.8	**44.0**	**49.0**	**52.2**	**60.1**	57.7	55.2
LSH	21.2	35.4	44.6	55.1	63.1	70.1	16.0	23.1	33.2	35.6	42.7	55.5
RandExp	**36.6**	**49.0**	49.5	**66.1**	**69.6**	70.9	29.3	35.8	41.4	50.4	**59.0**	**61.6**
ITQ	-17.5	-8.9	26.3	41.7	50.5	66.2	9.9	7.8	10.1	17.7	32.8	47.3
SH	4.5	5.8	20.3	42.9	61.3	**72.6**	18.9	17.6	27.5	35.45	48.1	57.9
PCAH	1.9	9.6	19.8	40.9	53.3	68.2	15.5	15.1	27.1	41.7	46.5	56.2

Table 7: Evaluation on word similarity datasets. For each dataset and hash length, the best (second best) score is in **bold** (underlined). The performance for GloVe embeddings is reported next to the name of each dataset in the format 300d/100d. Spearman's rank correlation coefficient is reported for common baselines that binarize GloVe (300d) embeddings together with our results. Hyperparameter settings for our algorithm: $K = 400$, $w = 11$.

Our aim here is to demonstrate that the fruit fly word embeddings are competitive with existing state-of-the-art binarization methods applied to GloVe and word2vec embeddings. We show this by evaluating the semantic similarity of static word embeddings, using several common benchmark datasets: WS353 (Finkelstein et al., 2002), MEN (Bruni et al., 2014), RW (Luong et al., 2013), SimLex (Hill et al., 2015), RG-65 (Rubenstein & Goodenough, 1965), and Mturk (Halawi et al., 2012). These datasets contain pairs of words with human-annotated similarity scores between them. Specif-

ically, we compare with GloVe (Pennington et al., 2014) word embeddings[2] trained on Wiki2014 and Gigaword 5, GloVe embeddings trained on OpenWebText Corpus (Gokaslan & Cohen, 2019) and word2vec embeddings[3].

Since our representations are binary (in contrast to GloVe and word2vec), we binarize GloVe and word2vec embeddings and report their performance using a number of common hashing methods, LSH/SimHash (Charikar, 2002) (random *contractive* projections followed by binarization based on sign), RandExp (Dasgupta et al., 2017) (random *expansive* projections followed by k-winner take all binarization), ITQ (Gong & Lazebnik, 2011) (iterative quantization), SH (spectral hashing) (Weiss et al., 2009), PCAH (Gong & Lazebnik, 2011) (PCA followed by binarization based on sign). Where available, we include evaluation from NLB, "Near-Lossless Binarization" (Tissier et al., 2019) (autoencoder-based binarization).

Following previous work (Tissier et al., 2019; Sokal, 1958), model similarity score for binary representations is evaluated as $sim(v_1, v_2) = (n_{11} + n_{00})/n$, where n_{11} (n_{00}) is the number of bits in v_1 and v_2 that are both 1 (0), and n is the length of $v_{1,2}$. Cosine similarity is used for real-valued representations. The results are reported in Tables 7, 8 and 9. For each dataset, we report performance across a range of hash lengths $\{4, 8, 16, 32, 64, 128\}$. For methods that incorporate randomness (LSH, RandExp, ITQ), we report the average across 5 runs. ITQ, SH and PCAH in Tables 7 and 8 were trained using the top 400k most frequent words. Table 9 compares our method to GloVe trained on OpenWebText (same dataset that our method is trained on) using the same vocabulary as our method uses.

Our binary word embeddings demonstrate competitive performance compared to published methods for GloVe and word2vec binarization, and our algorithm can learn meaningful binary semantic representations directly from raw text. Importantly, our algorithm does not require training GloVe or word2vec embeddings first before binarizing them.

Method	Hash Length (k)						Hash Length (k)					
	4	8	16	32	64	128	4	8	16	32	64	128
	MEN (75.5)						**WS353** (66.5)					
Ours	34.0	**49.9**	**55.9**	56.7	55.3	51.3	**43.2**	**52.1**	**55.3**	**57.4**	**60.3**	51.7
LSH	**35.5**	42.5	53.6	**63.4**	**68.4**	**72.2**	26.0	34.7	43.9	50.3	56.0	58.6
RandExp	24.2	34.6	45.8	57.5	66.1	71.7	23.5	34.3	37.3	48.0	57.6	**63.7**
ITQ	9.2	13.3	25.1	41.5	57.6	68.5	16.0	18.1	22.5	30.2	43.9	54.8
SH	7.2	15.8	31.3	46.9	62.3	69.4	3.3	9.6	22.7	34.1	50.0	54.7
PCAH	5.3	18.6	37.7	52.0	63.9	71.6	17.3	24.9	38.5	42.0	52.1	59.3
	SIMLEX (41.7)						**RW** (61.3)					
Ours	13.4	16.5	22.8	22.1	21.1	17.0	11.0	22.6	25.8	36.9	38.6	35.2
LSH	17.0	21.2	26.8	30.9	34.4	35.1	21.8	**27.8**	36.3	45.0	49.6	52.1
RandExp	**17.6**	**24.4**	**29.2**	**32.6**	**38.0**	**39.8**	**24.7**	27.7	**39.8**	**46.8**	**52.3**	**55.6**
ITQ	3.25	5.7	6.2	14.9	23.1	31.5	17.4	15.7	19.1	33.5	45.6	53.4
SH	-3.6	3.6	10.4	17.0	23.7	32.4	14.6	22.8	28.7	37.9	43.5	52.4
PCAH	-2.9	2.5	11.8	17.0	24.0	36.0	15.0	21.5	28.8	35.4	46.4	50.6
	RG (75.4)						**Mturk** (69.8)					
Ours	24.0	40.4	51.3	62.3	63.2	55.8	**44.0**	**49.0**	**52.2**	**60.1**	57.7	55.2
LSH	**44.6**	**55.1**	**63.1**	**70.1**	**76.4**	**75.8**	33.1	35.6	42.7	55.5	58.6	**62.4**
RandExp	30.4	42.0	48.6	59.1	70.2	74.6	22.7	34.8	42.0	45.9	57.9	61.2
ITQ	32.8	49.7	31.5	55.9	62.2	71.6	22.5	21.3	42.3	46.9	59.3	60.7
SH	18.0	30.6	36.0	48.8	56.9	**75.8**	21.9	27.4	41.8	51.2	58.8	58.0
PCAH	20.8	22.9	40.6	36.5	59.0	71.2	23.6	34.4	45.5	55.7	**64.2**	60.5

Table 8: Evaluation on word similarity datasets, analogous to Table 7, for 300d word2vec embeddings.

[2]pretrained embeddings: https://nlp.stanford.edu/projects/glove

[3]pretrained embeddings: https://code.google.com/archive/p/word2vec

Method	Hash Length (k)						Hash Length (k)					
	4	8	16	32	64	128	4	8	16	32	64	128
	MEN (76.4)						**WS353** (72.2)					
Ours	**34.0**	**49.9**	**55.9**	56.7	55.3	51.3	**43.2**	**52.1**	**55.3**	57.4	60.3	51.7
LSH	23.6	29.1	37.4	49.6	60.6	67.0	20.2	29.0	35.5	47.5	53.3	61.4
RandExp	28.4	40.3	52.3	**62.5**	**67.7**	71.0	30.5	40.0	48.1	**57.9**	63.3	67.5
ITQ	26.9	33.9	46.3	56.1	64.1	70.3	25.9	33.7	44.5	56.1	**63.9**	**67.6**
SH	23.8	28.7	44.1	54.7	62.1	69.7	18.1	25.7	40.1	51.8	60.9	62.9
PCAH	26.0	30.1	46.3	57.9	67.5	**72.4**	21.2	30.5	43.8	50.7	61.1	59.9
	SIMLEX (34.0)						**RW** (54.5)					
Ours	**13.4**	16.5	22.8	22.1	21.1	17.0	11.0	22.6	25.8	36.9	38.6	35.2
LSH	8.0	16.8	19.0	24.8	26.7	32.9	16.2	21.0	26.1	33.6	40.8	47.0
RandExp	10.1	**17.3**	**23.4**	**26.6**	29.7	31.3	22.0	**28.8**	34.1	**43.9**	46.3	**51.5**
ITQ	7.3	13.8	14.4	20.9	25.3	30.3	**24.5**	26.8	**34.8**	43.2	**49.1**	**51.5**
SH	12.1	14.2	17.5	20.0	26.4	36.0	19.7	24.8	32.9	38.7	45.4	46.7
PCAH	11.5	13.8	16.4	22.6	**31.1**	**38.6**	19.7	24.8	32.9	38.7	45.4	46.7
	RG (78.7)						**Mturk** (71.1)					
Ours	24.0	40.4	**51.3**	62.3	63.2	55.8	**44.0**	**49.0**	52.2	60.1	57.7	55.2
LSH	25.5	24.9	34.6	62.1	61.8	**73.5**	18.3	31.3	31.4	42.9	56.5	60.7
RandExp	28.7	45.6	47.3	**63.7**	**67.8**	70.8	28.3	41.3	50.1	56.5	65.4	67.1
ITQ	21.4	32.7	50.4	57.7	67.6	70.3	26.3	41.4	53.2	61.2	**67.1**	**68.9**
SH	39.8	45.6	50.0	50.2	62.3	68.6	20.3	35.9	51.9	61.9	59.1	61.3
PCAH	**45.0**	**50.0**	49.2	46.8	66.6	69.8	24.9	40.7	**55.7**	**64.3**	64.4	60.5

Table 9: Evaluation on word similarity datasets, analogous to Table 7. The 300d GloVe embeddings trained from scratch on the same OpenWebText dataset as our algorithm.

8 Appendix C. Details of Technical Implementation.

From the practical perspective, efficient implementation of the learning algorithm for the fruit fly network requires the use of sparse algebra, atomic updates, and block-sparse data access. Our algorithm is implemented in CUDA as a back-end, while python is used as an interface with the main functions.

The typical memory footprint of our approach is very small. About $100 - 270$MB GPU memory is allocated for the operators $W_{\mu i}$, $\mathbf{v^A}$ and temporary fields; while approximately 140GB CPU memory is needed to store the input data, array of random numbers for shuffle operations and shuffled indices. For GPU implementation, the model data is stored in the GPU's memory, while the input data is stored in the CPU memory. The parallelization strategy in our implementation is based on two aspects. First, each minibatch of data is divided into smaller sub-minibatches which are processed on different GPUs. Second, all the operations (dense-sparse matrix multiplications, $\arg\max$ operation, and weight updates) are executed in parallel using multiple threads.

9 Appendix D. Qualitative Evaluation of Contextual Embeddings.

In order to evaluate the quality of contextualized embeddings we have created an online tool, which we are planning to release with the paper, that allows users to explore the representations learned by our model for various inputs (context-target pairs). For a given query the tool returns the word cloud visualizations for each of the four top activated Kenyon cells. We show some examples of the outputs produced by this tool in Fig. 6. Each query is used to generate a bag of words input vector $\mathbf{v^A}$. This vector is then used to compute the activations of KCs using $\left\langle \mathbf{W}_\mu, \mathbf{v^A} \right\rangle$. Top four KCs with the highest activations are selected. The corresponding four weight vectors are used to generate four probability distributions of individual words learned by those KCs by passing the weights through a softmax function. For example, for one of those vectors with index μ, the probability distribution is computed as $\text{prob}_i = SM(W_{\mu i})$. These probability distributions for the top four activated KCs are visualized as word clouds. In computing the softmax only the target block of the weight vector was used (we have checked that using only the context block gives qualitatively similar word clouds).

Figure 6: Examples of three queries and corresponding word cloud visualization for top four activated KCs (by each query).

The results indicate that the fruit fly network indeed has learned meaningful representations. Consider for example the first query. The sentence: "Entertainment industry shares rise following the premiere of the mass destruction weapon documentary" results in the four top activated KCs shown in Fig. 6. The top activated KC has the largest weights for the words "weapon", "mass", etc. The second activated KC is sensitive to the words "market", "stock", etc. This illustrates how the fruit fly network processes the queries. In this example the query refers to several distinct combinations of concepts: "weapon of mass destruction", "stock market", "movie industry". Each of those concepts has a dedicated KC responsible for it. As one can see the responses are not perfect. For example in this case one would expect to have the 4-th highest activated KC, which is responsible for the "movie industry" concept to have a higher activation than the 3-rd highest KC, which is responsible for the types of "weapons of mass destruction". But overall all the concepts picked by the KCs are meaningful and related to the query.

10 Appendix E. Details of GloVe Retraining

To directly compare our method to GloVe, we trained an uninitialized GloVe model on the same OpenWebText corpus using the code provided by the original GloVe authors (Pennington et al., 2014)[4]. This model was optimized to have the same vocab size as our model (the 20k most frequent tokens), used an embedding size of 300, and a window size of 15. The model was trained for 180 iterations at about 3 minutes, 20 seconds per iteration on 16 threads, resulting in the total training time of approximately 10 hours.

[4] https://nlp.stanford.edu/projects/glove/

11 APPENDIX F. RELATED WORK.

Our work builds on several ideas previously discussed in the literature. The first idea is that fruit fly olfactory network can generate high quality hash codes for the input data in random (Dasgupta et al., 2017) and data-driven (Ryali et al., 2020) cases. There are two algorithmic differences of our approach compared to these previous studies. First, our network uses representational contraction, rather than expansion when we go from the PN layer to the KCs layer. Second, (Dasgupta et al., 2017; Ryali et al., 2020) construct hash codes for data coming from a single modality (e.g., images, or word vectors), while the goal of the present paper is to learn correlations between two different "modalities": target word and its context. The second idea pertains to the training algorithm for learning the PN$\rightarrow$KCs synapses. We use a biologically plausible algorithm of (Krotov & Hopfield, 2019) to do this, with modifications that take into account the wide range of frequencies of different words in the training corpus (we discuss these differences in section 2.1). Also, similarly to (Dasgupta et al., 2017; Ryali et al., 2020) the algorithm of (Krotov & Hopfield, 2019) is used for learning the representations of the data, and not correlations between two types of data (context and target) as we do in this paper.