

Recommendation system for scientific tools and workflows

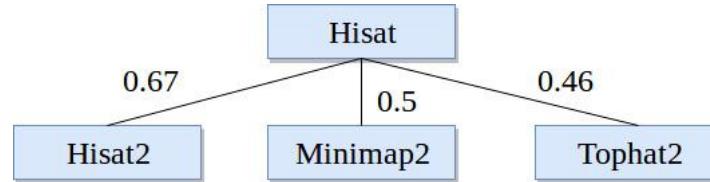
Master's thesis

Anup Kumar

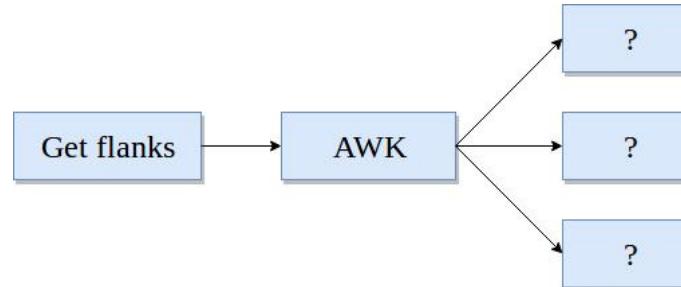
Adviser: Dr. Björn Grüning

Recommendation system

1. Find similar scientific tools



2. Predict tools for workflows



1. Find similar scientific tools

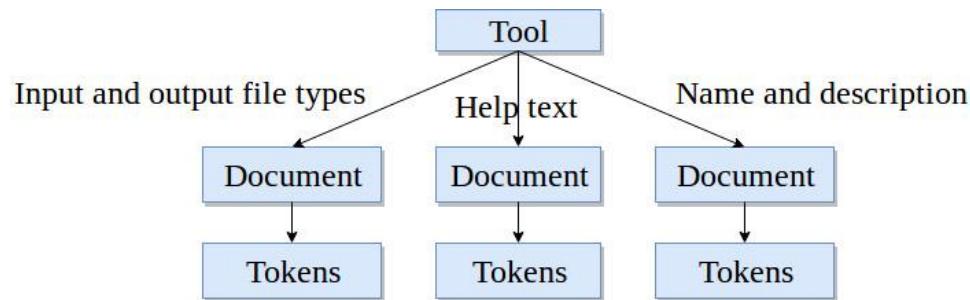
- Tools similar in functions
- Compute similarity among tools
- Recommend similar tools
- Replace a tool by its similar tool
- Provide more options for data processing

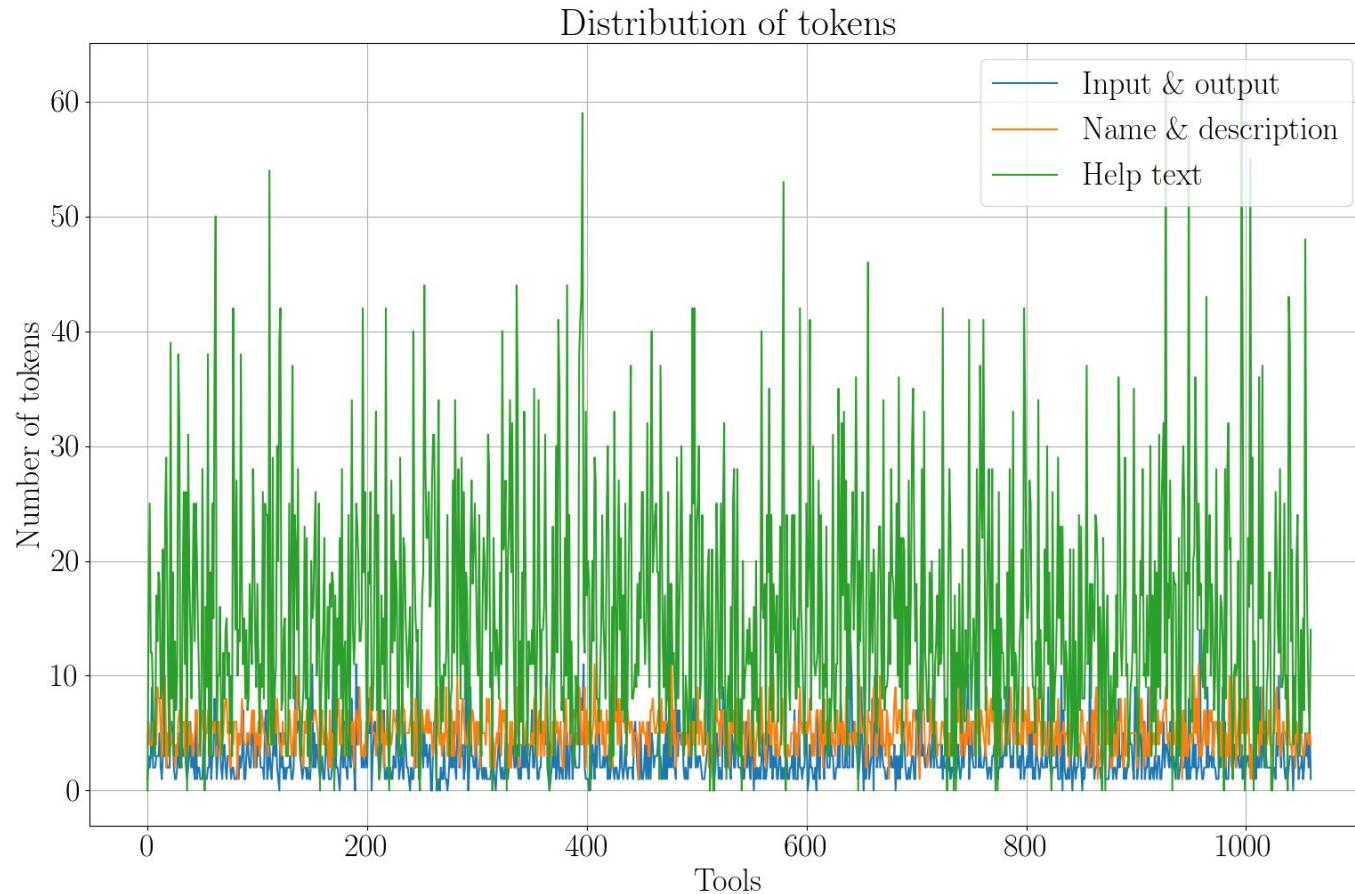
Approach

1. Collect tools metadata (~ 1,050 tools)
2. Clean metadata - stemming and stopwords
3. Learn vectors for tools
4. Compute similarity among vectors
5. Combine similarity matrices

Tool, documents and tokens

- Three attributes
- Document - a collection of tokens





Paragraph (document) vector

- Neural network approach*
- A dense vector for each paragraph
- Similar tools, similar vectors
- Encode variable length paragraphs
- Each tool has three paragraphs (documents)

*https://cs.stanford.edu/~quocle/paragraph_vector.pdf

Input/output file types - <https://dl.acm.org/citation.cfm?id=1704810> (bestmatch25)

Similarity scores

- Jaccard index (input and output file types)
- Cosine angle (name and description and help text)
- Compute similarity matrix
- Three similarity matrices
- Simple - average the matrices
- Better - learn weights

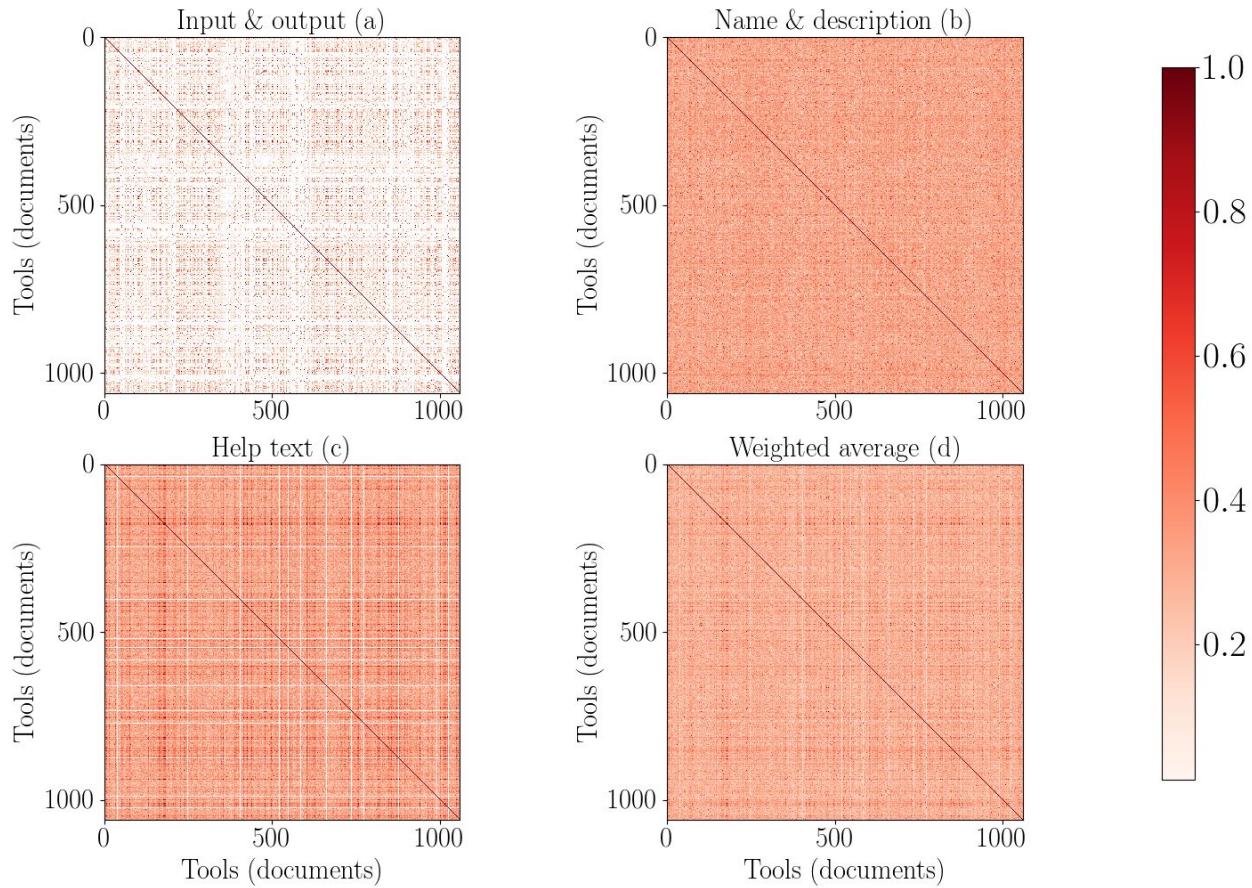
$$j = \frac{A \cap B}{A \cup B}$$

$$x \cdot y = |x| \times |y| \times \cos \theta$$

Jaccard index: <https://link.springer.com/article/10.1007/BF02365362>

Cosine angle similarity: <https://www.itl.nist.gov/div898/software/dataplot/refman2/auxiliar/cosdist.htm>

Similarity matrices



Optimisation

- Gradient descent
- Learn weights on similarity scores

$$Error(w^k) = \frac{1}{N} \times \sum_{j=1}^N [w^k \times SM^k - SM_{ideal})^2]_j \quad \dots (1)$$

$$Gradient(w^k) = \frac{\partial Error}{\partial w^k} = \frac{2}{N} \times ((w^k \times SM^k - SM_{ideal}) \cdot SM^k) \quad \dots (2)$$

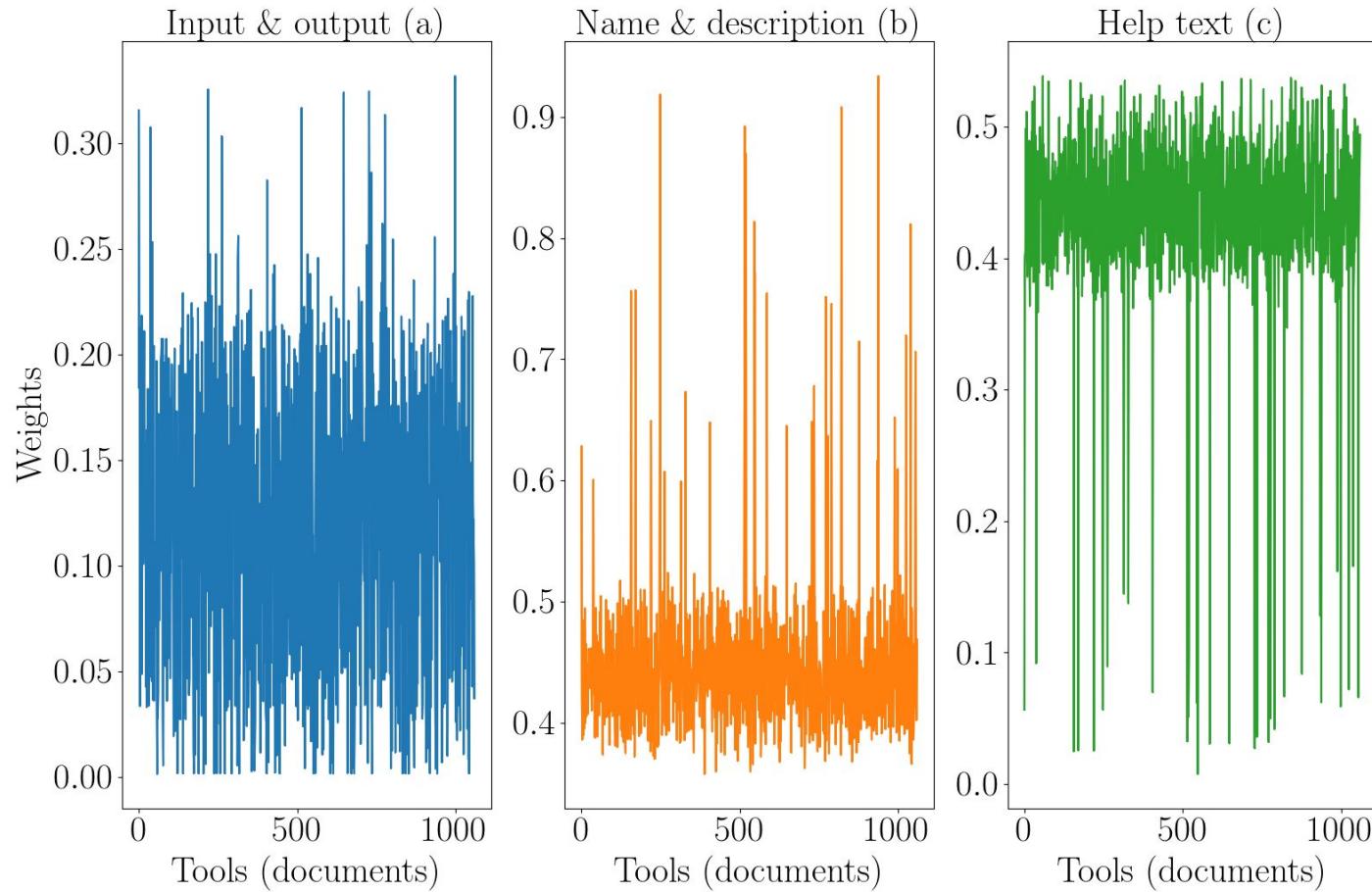
$$w^k = w^k - \eta \times Gradient(w^k) \quad \dots (3)$$

- Obtain a weighted average similarity matrix

SM_{ideal} is an array of 1.0 as the maximum value of similarity measures can be at most 1.0

Gradient descent: <https://developers.google.com/machine-learning/crash-course/reducing-loss/gradient-descent>

Distribution of weights

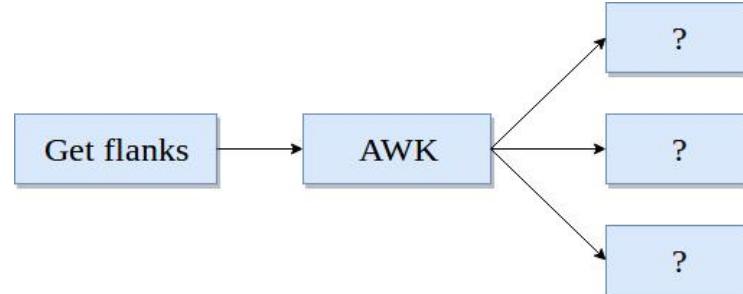


Summary and conclusion

- Collect and clean tools metadata (~ 1,050 tools)
- Learn vector for each tool
- Compute and combine similarity matrices
- Less data for name and description attribute
- No true similarity
- Exclude low similarity values
- Run analysis on larger set of tools
- Compute similar tools using workflows

2. Predict tools in workflows

- Thousands of tools
- Tough to assemble workflows
- Tools compatibility issues
- Loss of computation time

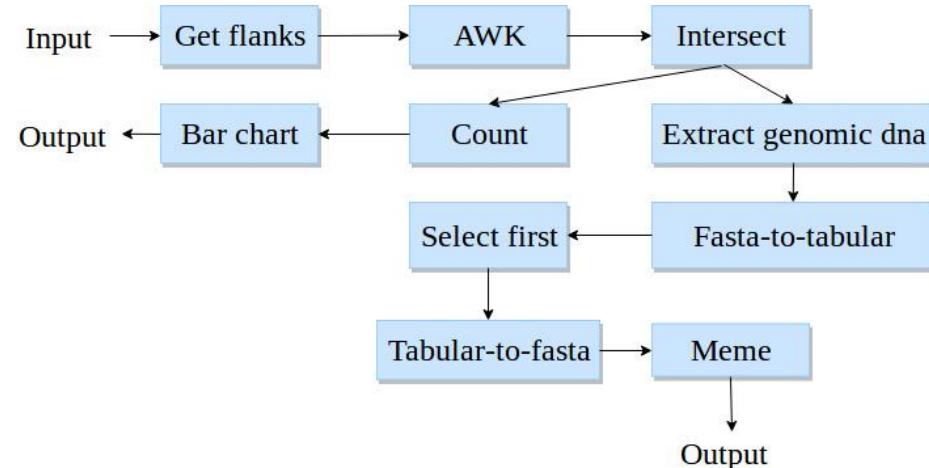


Approach

- Extract paths from workflows
- Remove duplicate paths
- Preprocess the paths
- Divide into training and test sets
- Classify using recurrent neural network
- Compute precision

Workflows

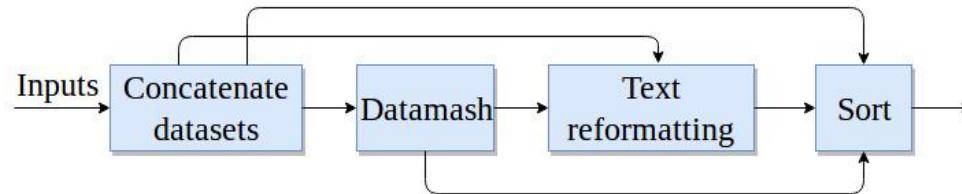
- Workflow - a directed acyclic graph
- Paths in a workflow



- 193,000 workflows
- 900,000 paths (167,000 unique)
- Maximum 25 tools in a path

Higher order dependency

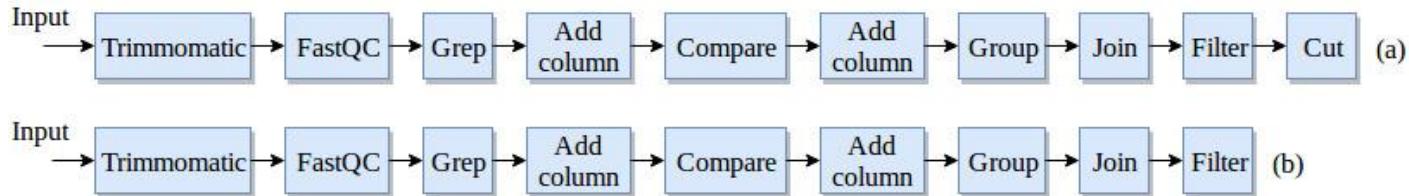
- Not dependent only on immediate parent
- Dependent on all previous tools



Higher order dependency: <https://arxiv.org/pdf/1506.06268.pdf>
<https://arxiv.org/pdf/1508.03113.pdf>

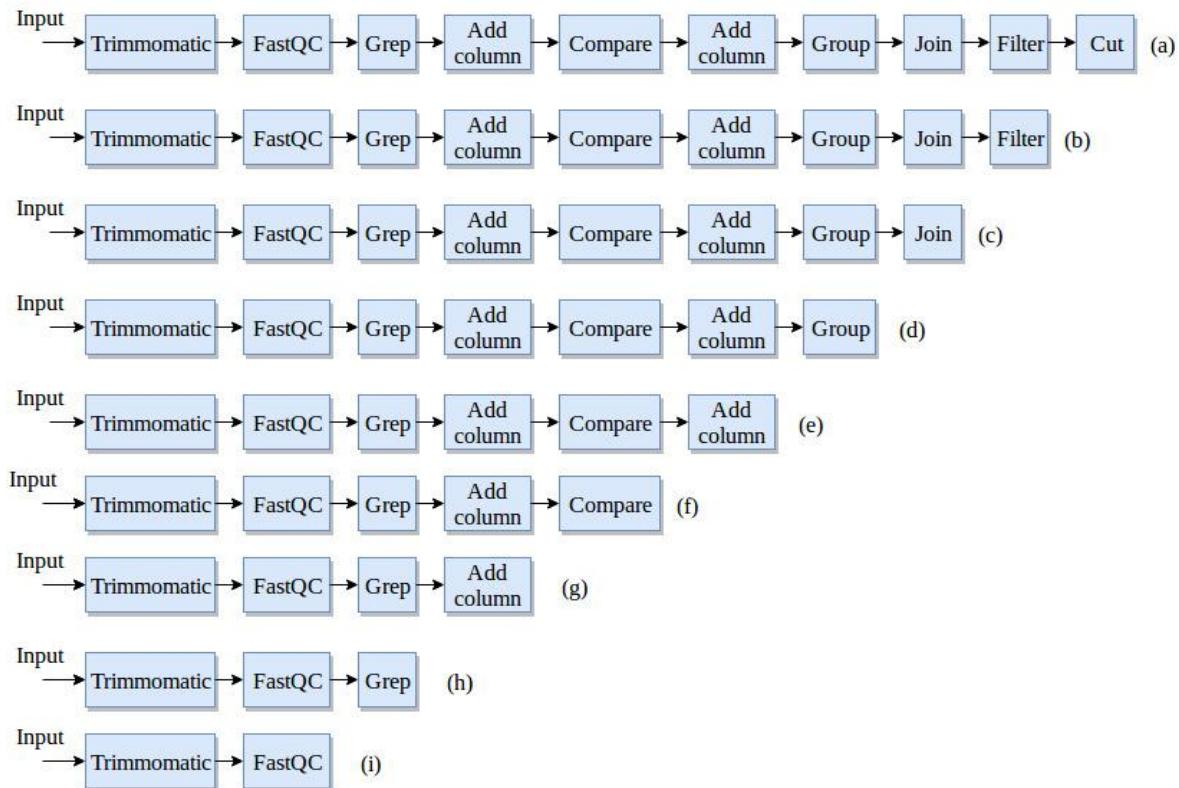
Workflow preprocessing

- No decomposition
- Last tool is a label



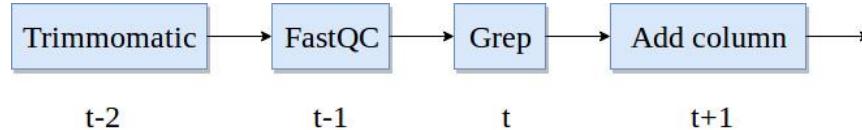
Workflow preprocessing

- Decomposition
- Last tool is label



Recurrent neural network

- Classifier to learn on sequential data
- Multilabel, multiclass classification
- Recurrent neural network - Gated recurrent units
- Learn long-range dependencies



Recurrent neural network: <https://arxiv.org/pdf/1412.3555.pdf>

Gated recurrent units: <https://arxiv.org/pdf/1412.3555v1.pdf>

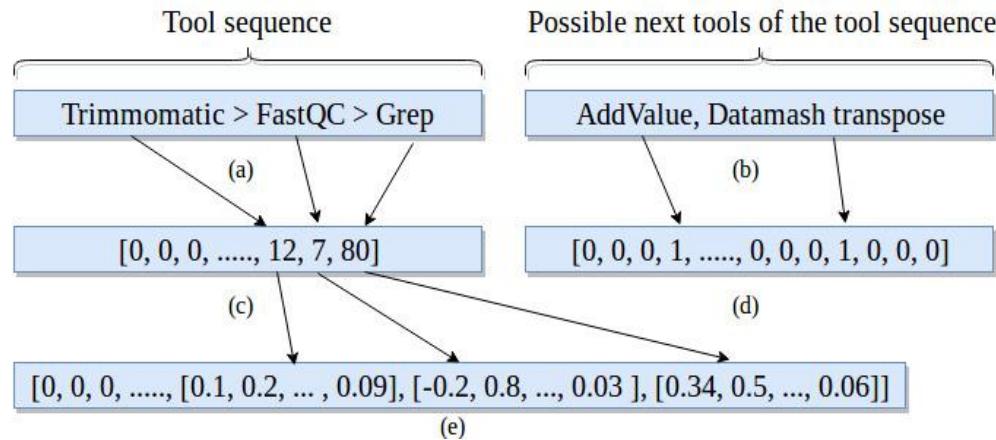
Network configuration

- Two hidden layers, one embedding layer and one output layer
- Cross-entropy loss
- Root mean square propagation (rmsprop) optimiser
- Sigmoid output activation

$$f(x) = \frac{1}{1+e^{-x}}$$

- 80% training paths, 20% test paths
- 20% of training paths as validation paths

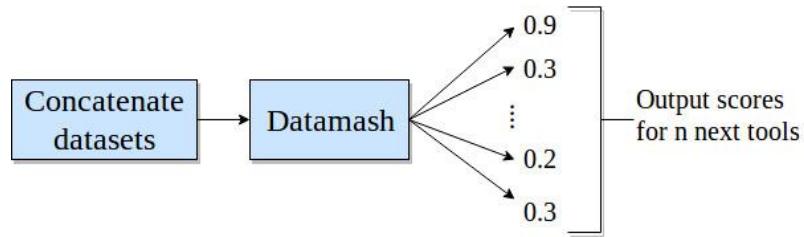
Embedding and label vectors



- Dimensions:
 - Embedding - 512
 - Label vector - 1,800
 - Tool sequence - 25

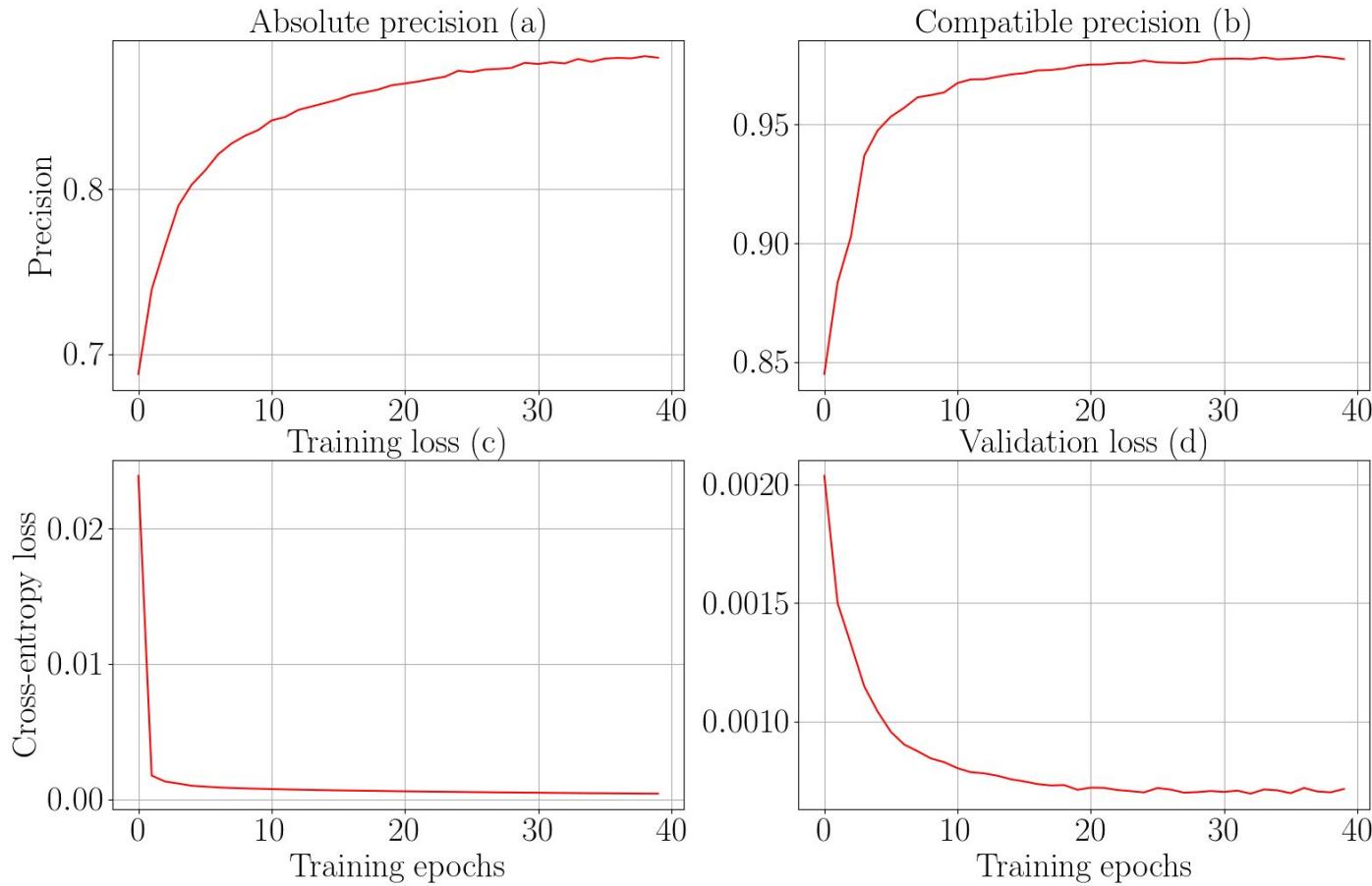
Prediction and precision

- Score for each predicted tool

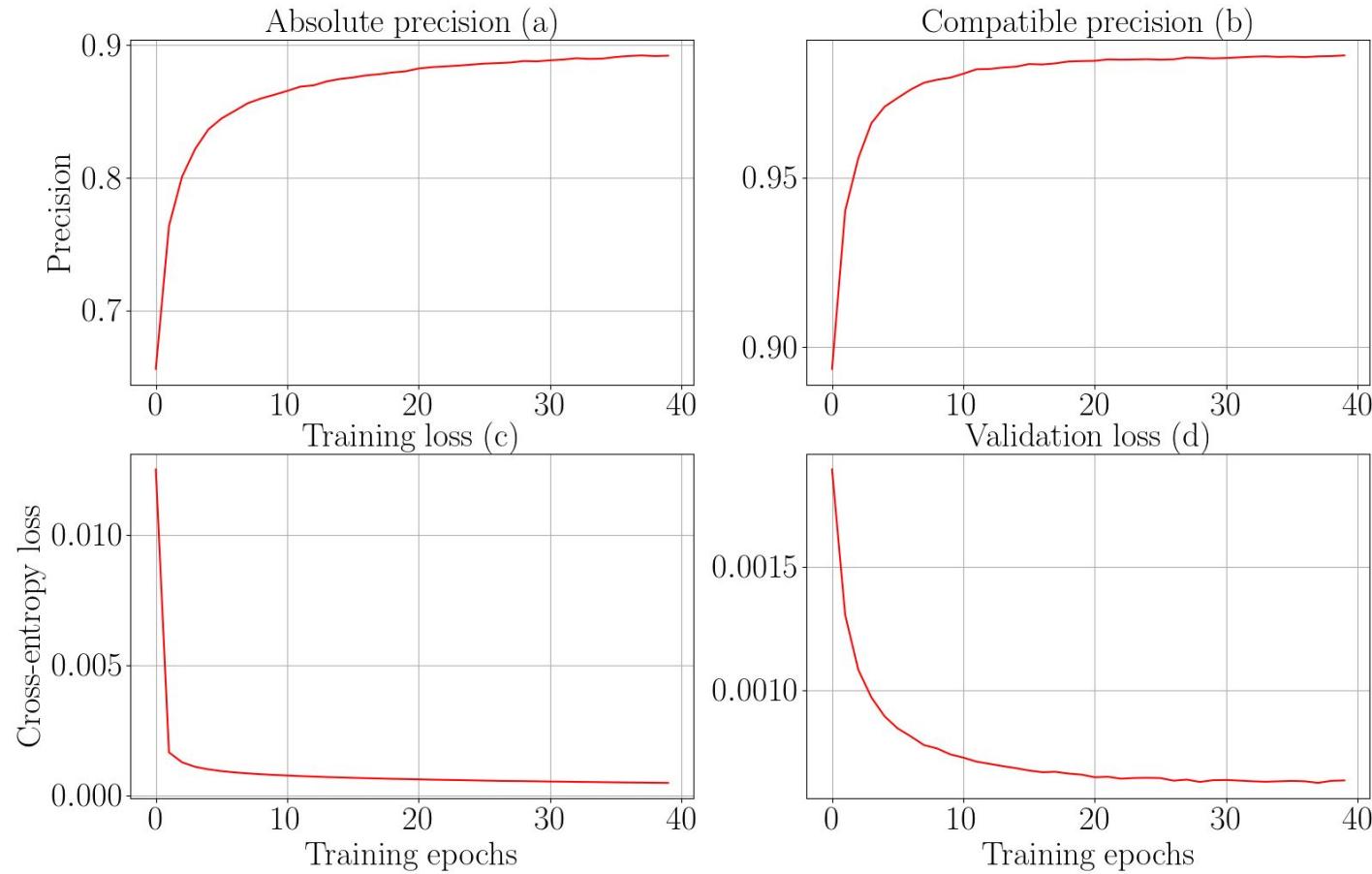


- Multiple possible tools for a path
- Absolute precision
- Compatible precision

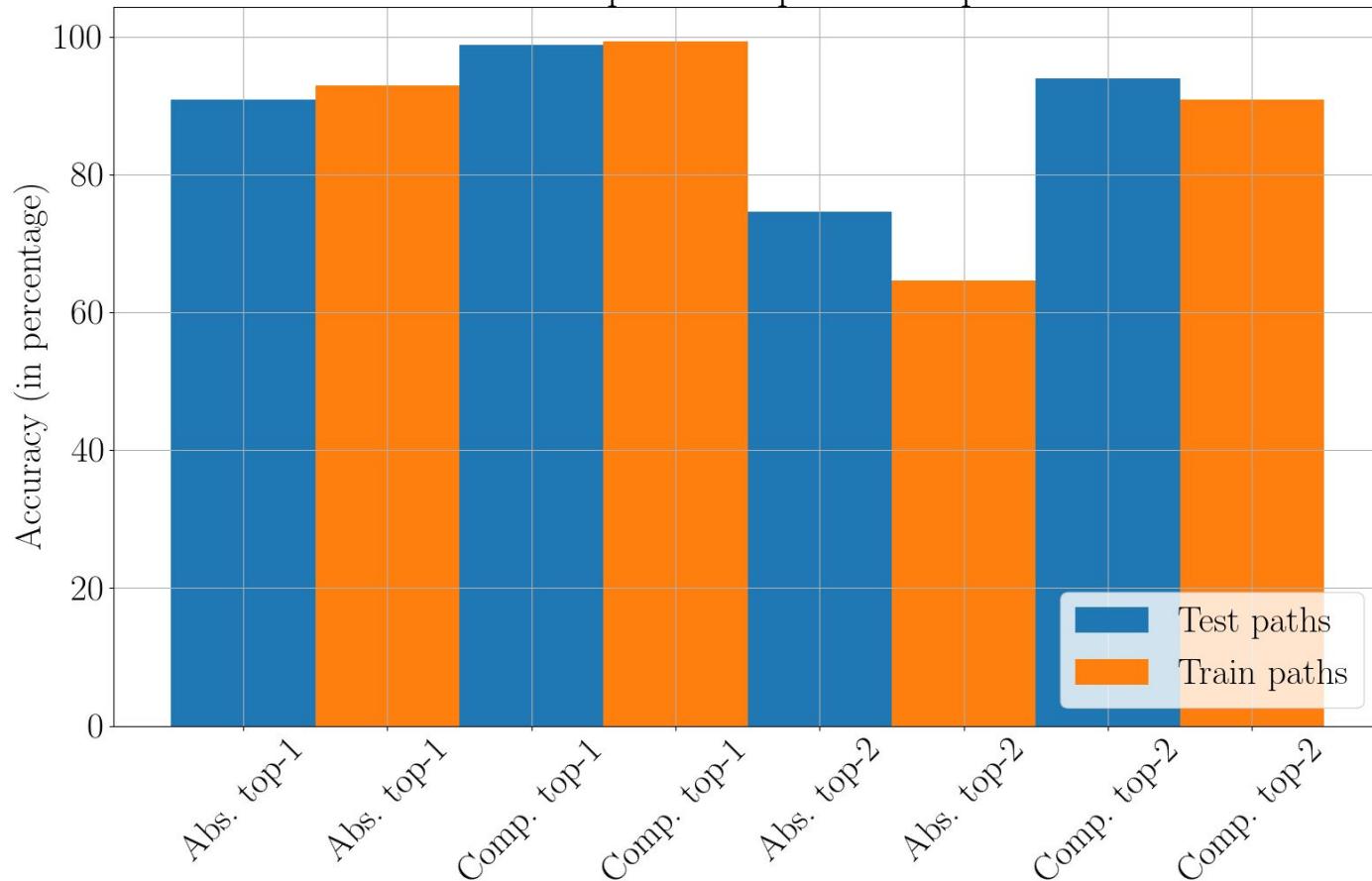
Precision and loss for no decomposition of paths



Precision and loss for decomposition of train and test paths



Absolute and compatible top-1 and top-2 accuracies



Summary and conclusion

- Workflows - directed acyclic graphs (193,000)
- Extract paths (167,000 unique) from workflows
- Recurrent neural network (gated recurrent units)
- Learn long-range dependencies
- Multilabel, multiclass classification
- No decomposition:
 - Absolute precision ~ 88%, compatible precision ~ 98% (~ 22 hrs)
- Decomposition:
 - Absolute precision ~ 89%, compatible precision ~ 99% (~ 48 hrs)
- More workflows, better precision
- Recommendation system using similar and predicted tools

Future work

- Restore original distribution
- Decay prediction over time
- Oversampling - data augmentation
- Integrate into Galaxy

Thank you for your attention

Questions?

Supplementary material

Stemming and stopwords

- Stemming - converge all forms of a word into one basic form
- “*Operate, operating, operates, operation, operative, operatives, operational*” into “*oper*” [1]
- Stopwords - “*a, about, above, would, could ...*” [2]

1. <https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html>
2. <https://www.ranks.nl/stopwords>

Bestmatch25 (bm25)

- Token frequency (tf)
- Document and inverted document frequency (idf)

$$idf = \log \frac{N}{df}$$

$$\alpha = (1 - b) + \frac{b \cdot |D|}{|D|_{avg}}$$

$$tf^* = tf \cdot \frac{k+1}{k \cdot \alpha + tf}$$

$$bm25 = tf^* \cdot idf$$

<https://dl.acm.org/citation.cfm?id=1704810>

http://www.staff.city.ac.uk/~sb317/papers/foundations_bm25_review.pdf

Bestmatch25 (bm25) scores

Tools/Tokens	Regress	Linear	Gap	Mapper	Perform
LinearRegression	5.22	4.1	0.0	0.0	3.84
LogisticRegression	3.54	0.0	0.0	0.0	2.61
Tophat2	0.0	0.0	1.47	1.47	0.0
Hisat	0.0	0.0	0.0	0.0	0.0

Latent Semantic Analysis

- Document-token matrix (X)
- Singular value decomposition

$$X_{n \times m} = U_{n \times n} \cdot S_{n \times m} \cdot V_{m \times m}^T$$

$$U^T \cdot U = I_{n \times n}$$

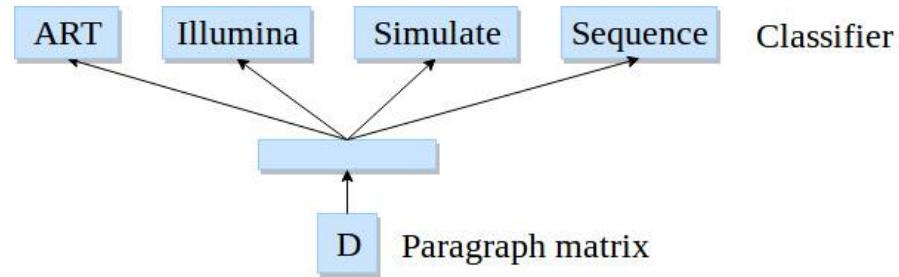
$$V^T \cdot V = I_{m \times m}$$

$$X_{n \times m} = U_k \cdot S_k \cdot V_k^T$$

Paragraph vectors

- Softmax classifier
- Backpropagation
- Stochastic gradient descent
- Gensim*
- 800 iterations, 10 epochs

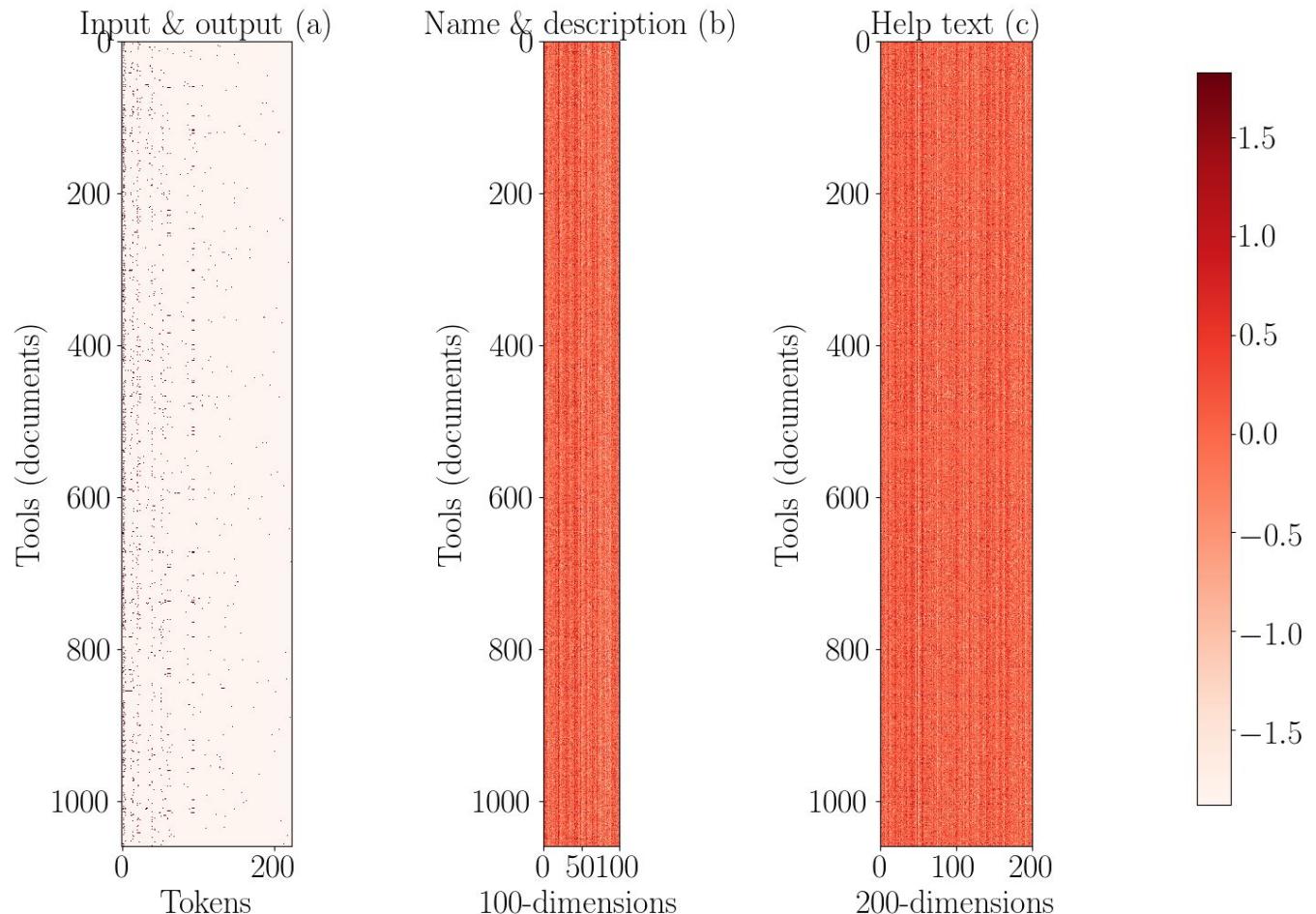
$$\frac{1}{T} \cdot \sum_{t=k}^{T-k} \log p(w_t | w_{t-k}, \dots, w_{t+k})$$



*<https://radimrehurek.com/gensim/models/doc2vec.html>

Image adapted from: https://cs.stanford.edu/~quocle/paragraph_vector.pdf

Document-token and paragraph matrices



Optimisation

- Gradient descent
- Mean squared error
- Initial learning rate - 0.05
- Decay learning rate
- Nesterov's accelerated gradient

$$update_{t+1} = \gamma \cdot update_t - \eta \cdot Gradient(w_t + \gamma \cdot update_t)$$

$$w_{t+1} = w_t + update_{t+1}$$

Optimisation

1	0.34	0.65	0.44
0.34	1
0.65	...	1	...
0.44	1

1	0.76	0.63	0.85
0.76	1
0.63	...	1	...
0.85	1

1	0.06	0.1	0.17
0.06	1
0.1	...	1	...
0.17	1

Optimisation

$$Error_{io}(w_{io}^k) = \frac{1}{N} \times \sum_{j=1}^N [(w_{io}^k \times SM_{io}^k - SM_{ideal})^2]_j$$

$$Error_{nd}(w_{nd}^k) = \frac{1}{N} \times \sum_{j=1}^N [(w_{nd}^k \times SM_{nd}^k - SM_{ideal})^2]_j$$

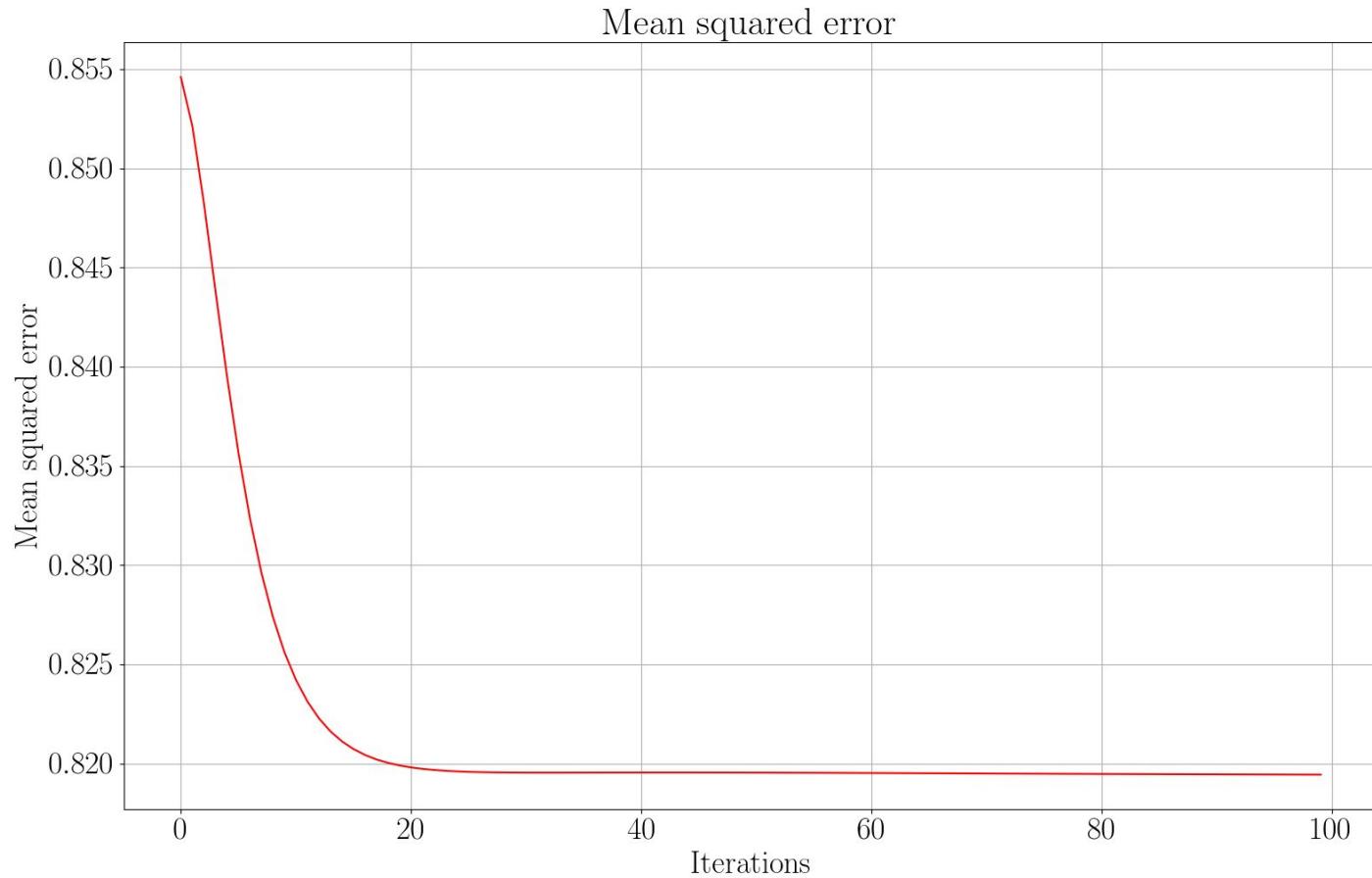
$$Error_{ht}(w_{ht}^k) = \frac{1}{N} \times \sum_{j=1}^N [(w_{ht}^k \times SM_{ht}^k - SM_{ideal})^2]_j$$

$$Error(w^k) = Error_{io}(w_{io}^k) + Error_{nd}(w_{nd}^k) + Error_{ht}(w_{ht}^k)$$

$$\operatorname{argmin}_{w^k} Error(w^k)$$

$$w_{io}^k + w_{nd}^k + w_{ht}^k = 1$$

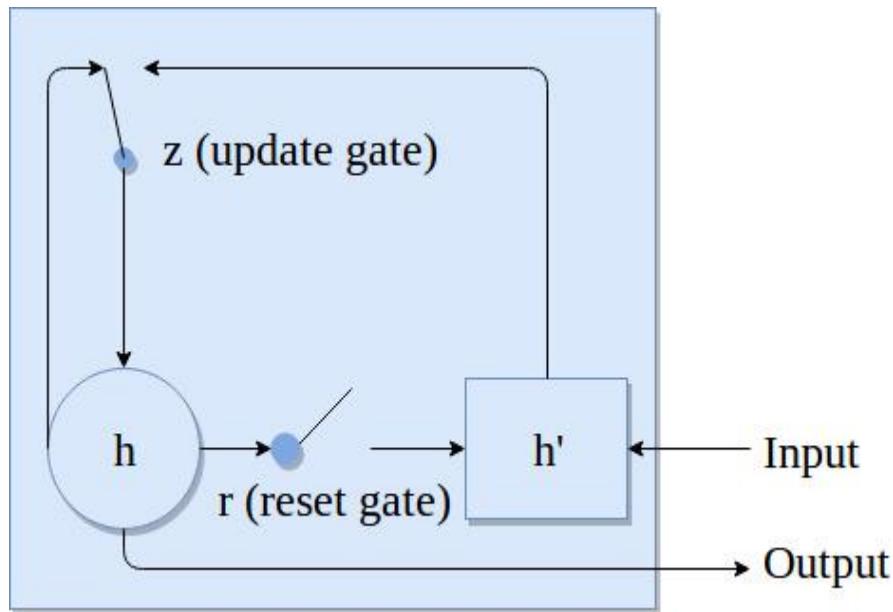
$$SM^k = w_{io}^k \cdot SM_{io}^k + w_{nd}^k \cdot SM_{nd}^k + w_{ht}^k \cdot SM_{ht}^k$$



Visualisers

- Paragraph vectors
- Latent Semantic Analysis (5% of full-rank)

Recurrent neural network



$$h_t = (1 - z_t) \times h_{t-1} + z_t \times h'_t$$

$$z_t = \sigma(W_z \times x_t + U_z \times h_{t-1})$$

$$r_t = \sigma(W_r \times x_t + U_r \times h_{t-1})$$

$$h'_t = \tanh(W \times x_t + U \times (r_t \odot h_{t-1}))$$

$$p(x_T | x_1, x_2, \dots, x_{T-1})$$

Recurrent neural network

- One embedding layer, two hidden layer and one output layer
- Recurrent layer activation - exponential linear unit

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha \times (e^x - 1), & \text{if } x \leq 0, \alpha > 0 \end{cases}$$

- Output activation is sigmoid $f(x) = \frac{1}{1+e^{-x}}$

Recurrent neural network

- Optimiser - Root mean square propagation (rmsprop)

$$MeanSquare(w, t) = 0.9 \times MeanSquare(w, t - 1) + 0.1 \times (\frac{\partial E}{\partial w}(t))^2$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{MeanSquare(w, t) + \epsilon}} \times \frac{\partial E}{\partial w}(t)$$

- Binary cross-entropy loss

$$loss_{mean} = -\frac{1}{N} (\sum_{i=1}^N y_i \times \log(p_i) + (1 - y_i) \times \log(1 - p_i))$$

Paths statistics

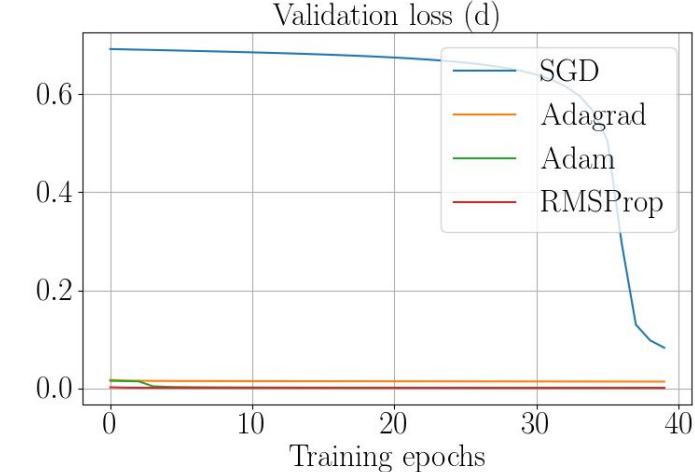
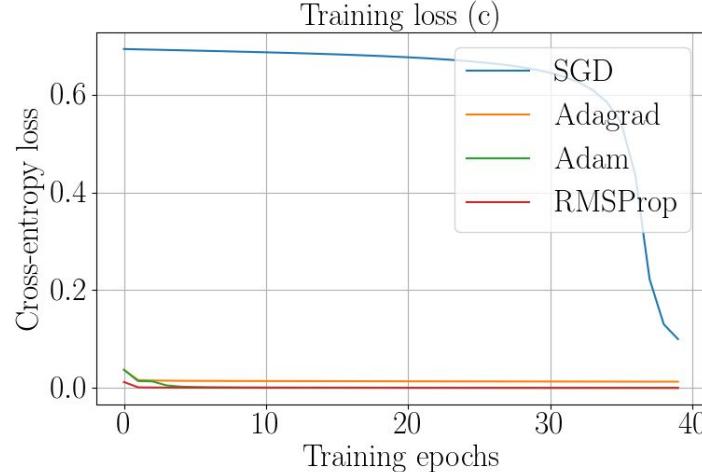
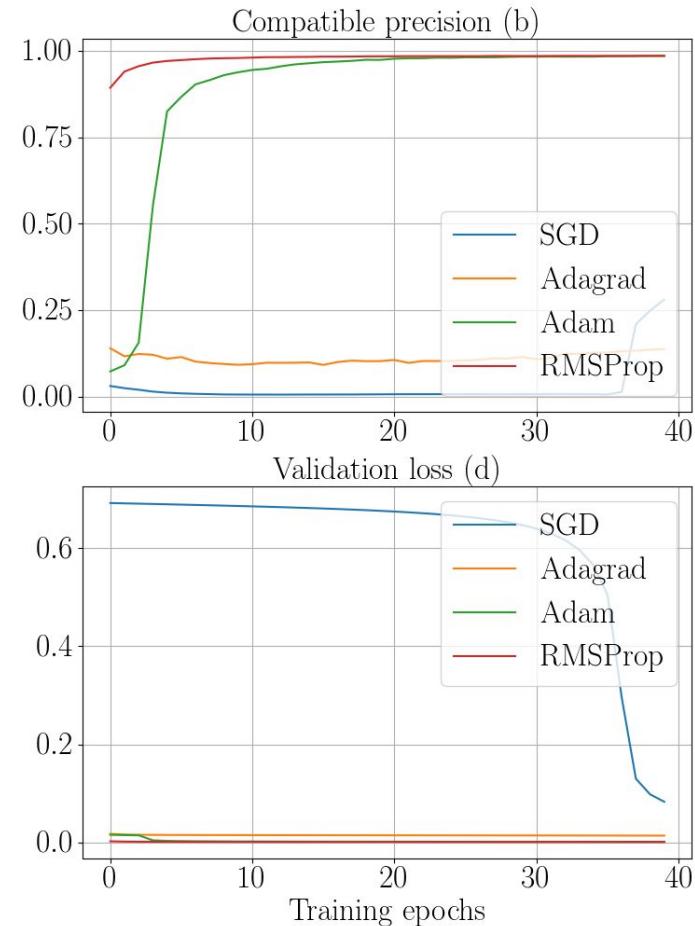
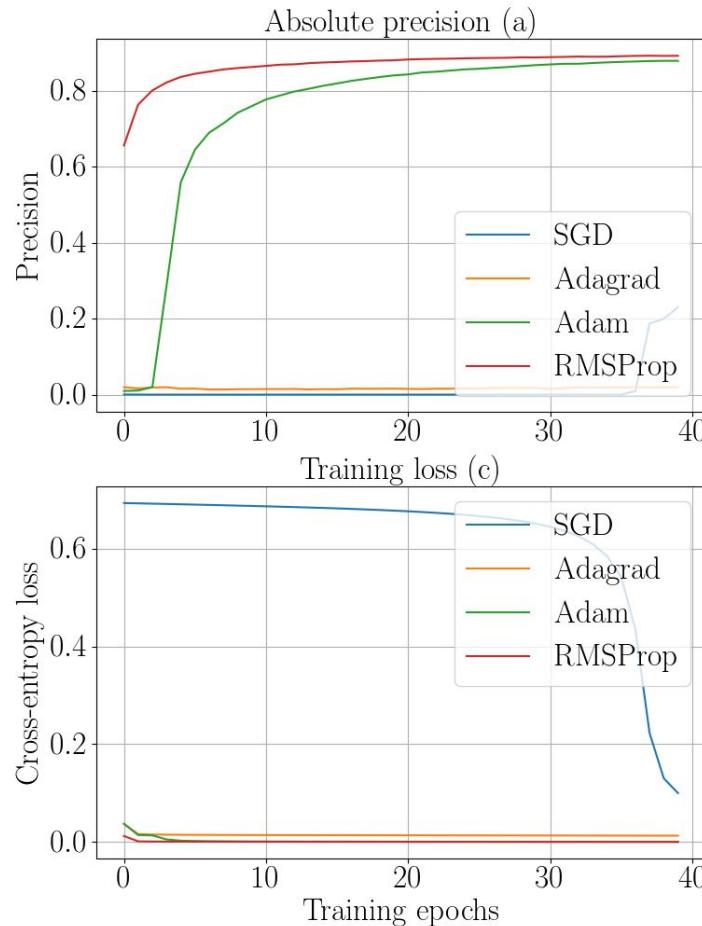
- No decomposition:

Total paths (111,386), train paths (89,109) and test paths (22,277).

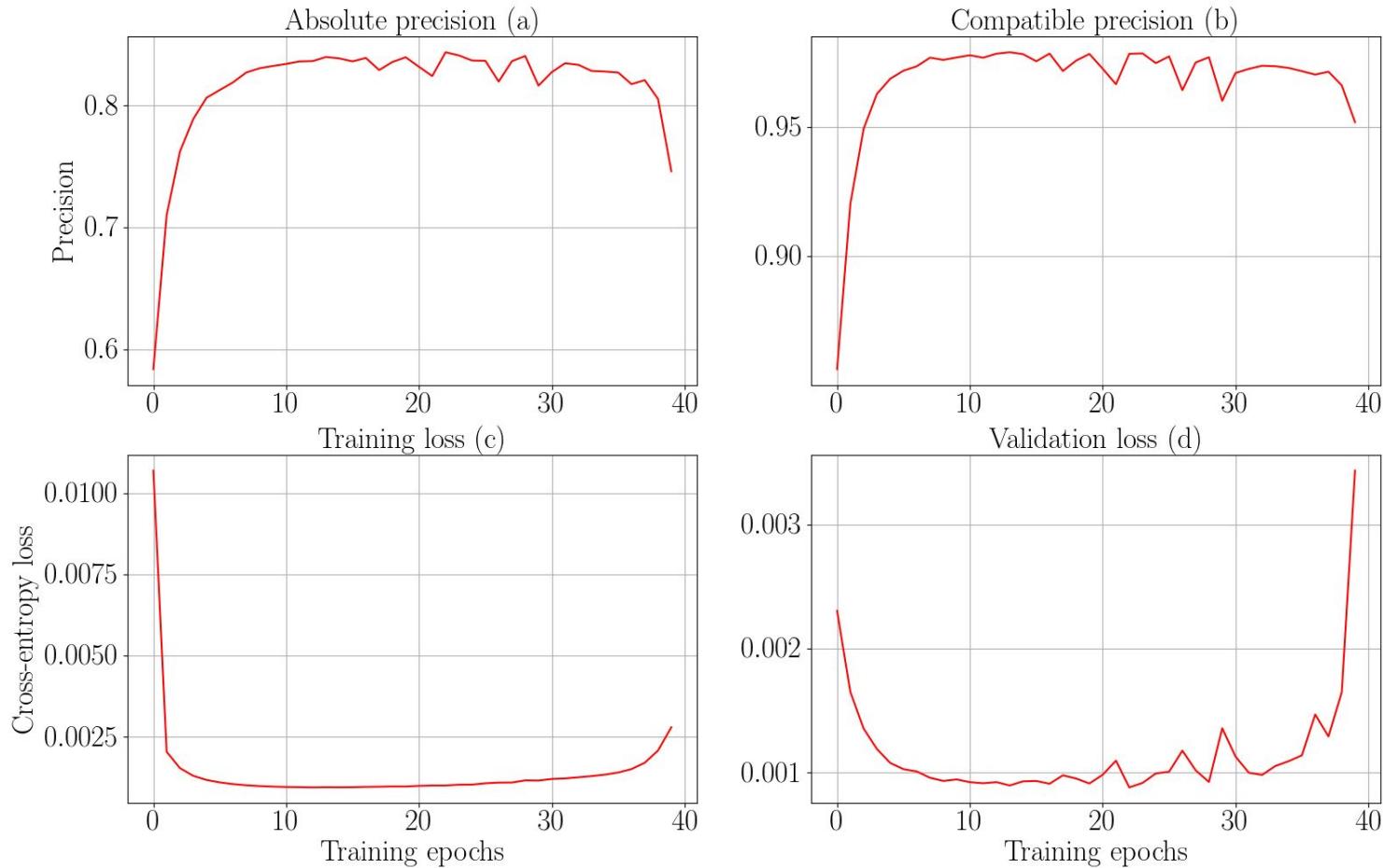
- Decomposition:

Total paths (210,983), train paths (168,787) and test paths (42,196).

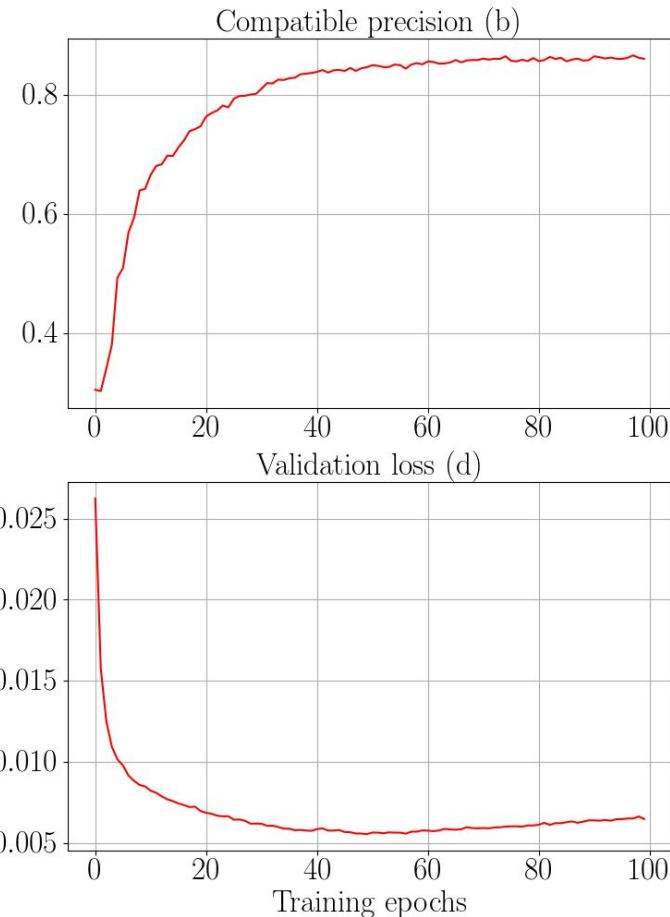
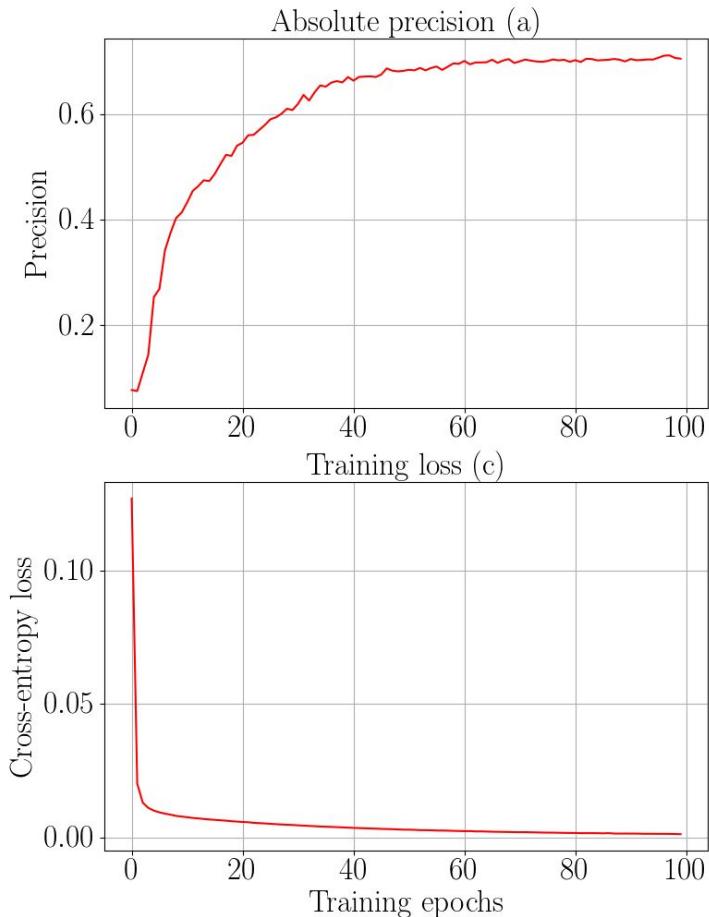
Precision and loss for various optimisers



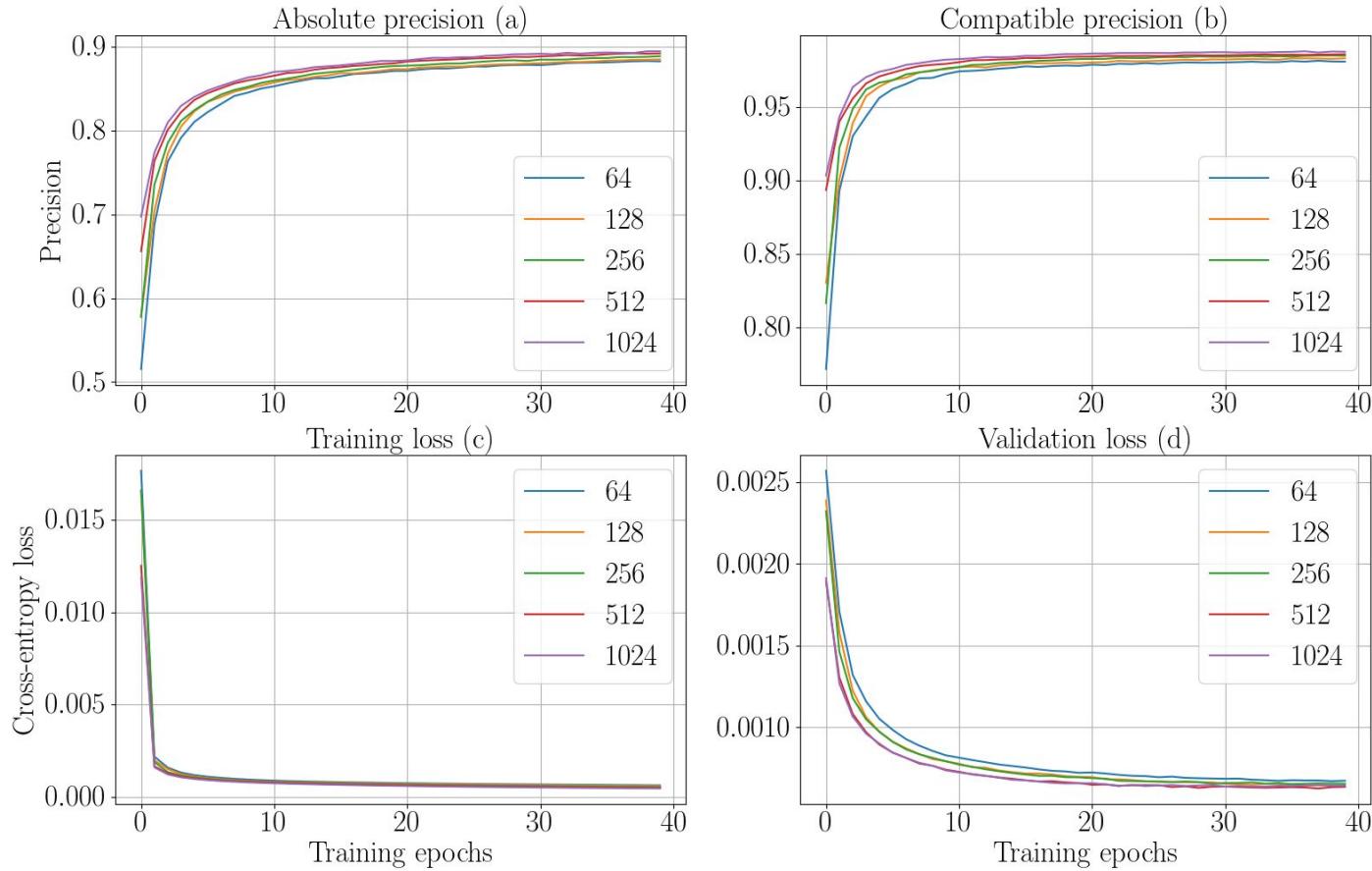
Precision and loss using neural network with dense layers



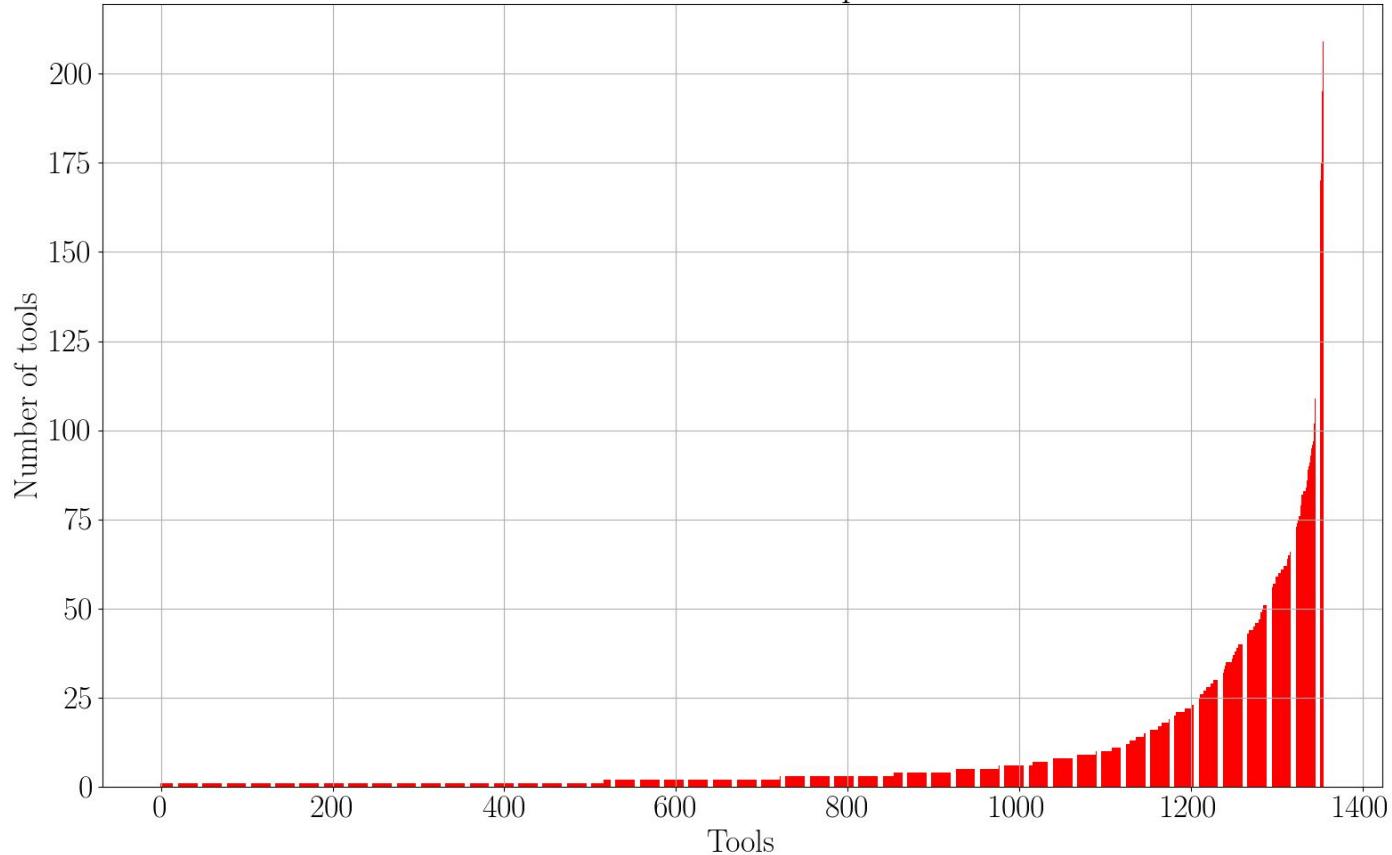
Precision and loss using less data



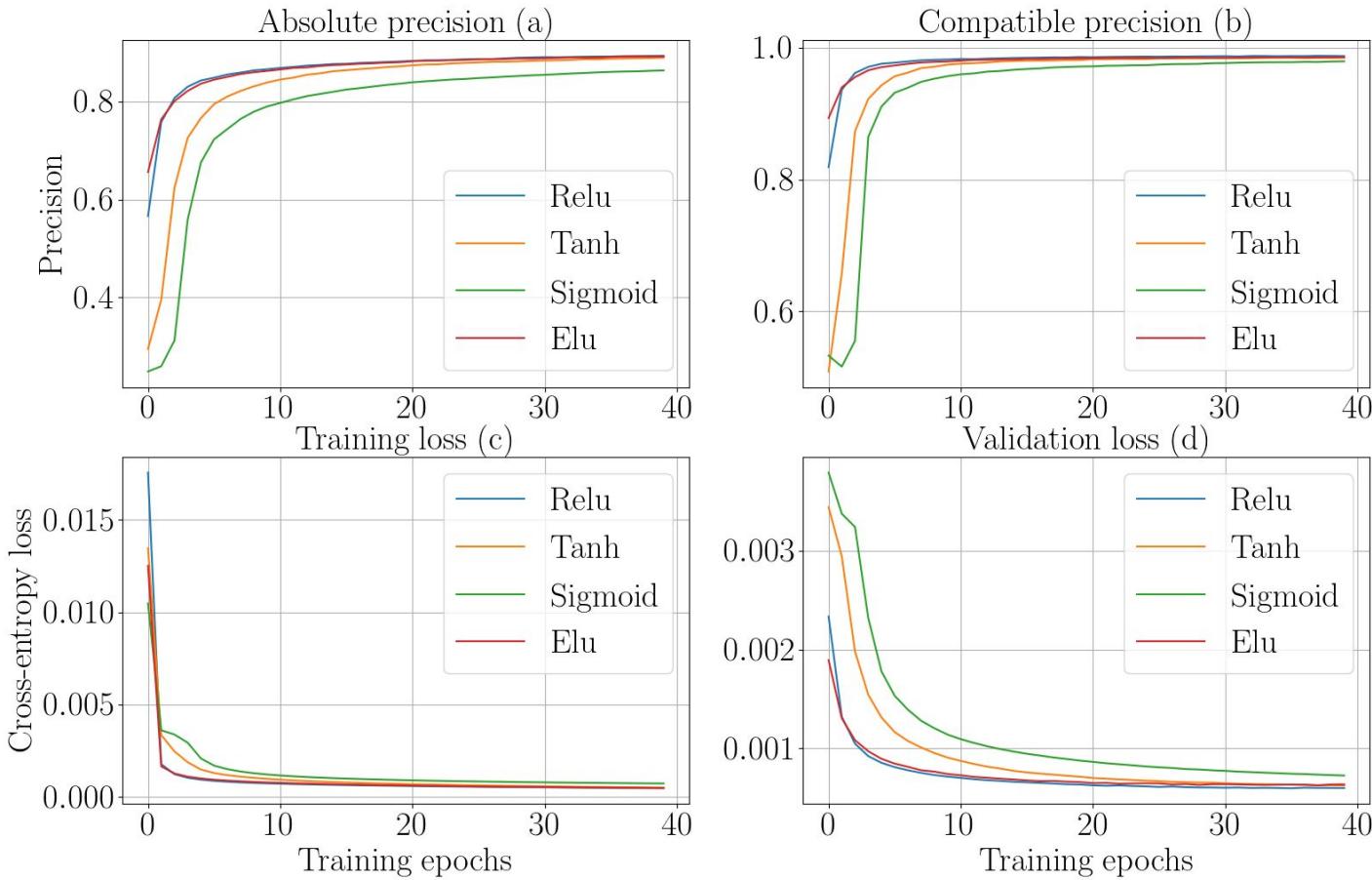
Precision and loss for various sizes of embedding layer



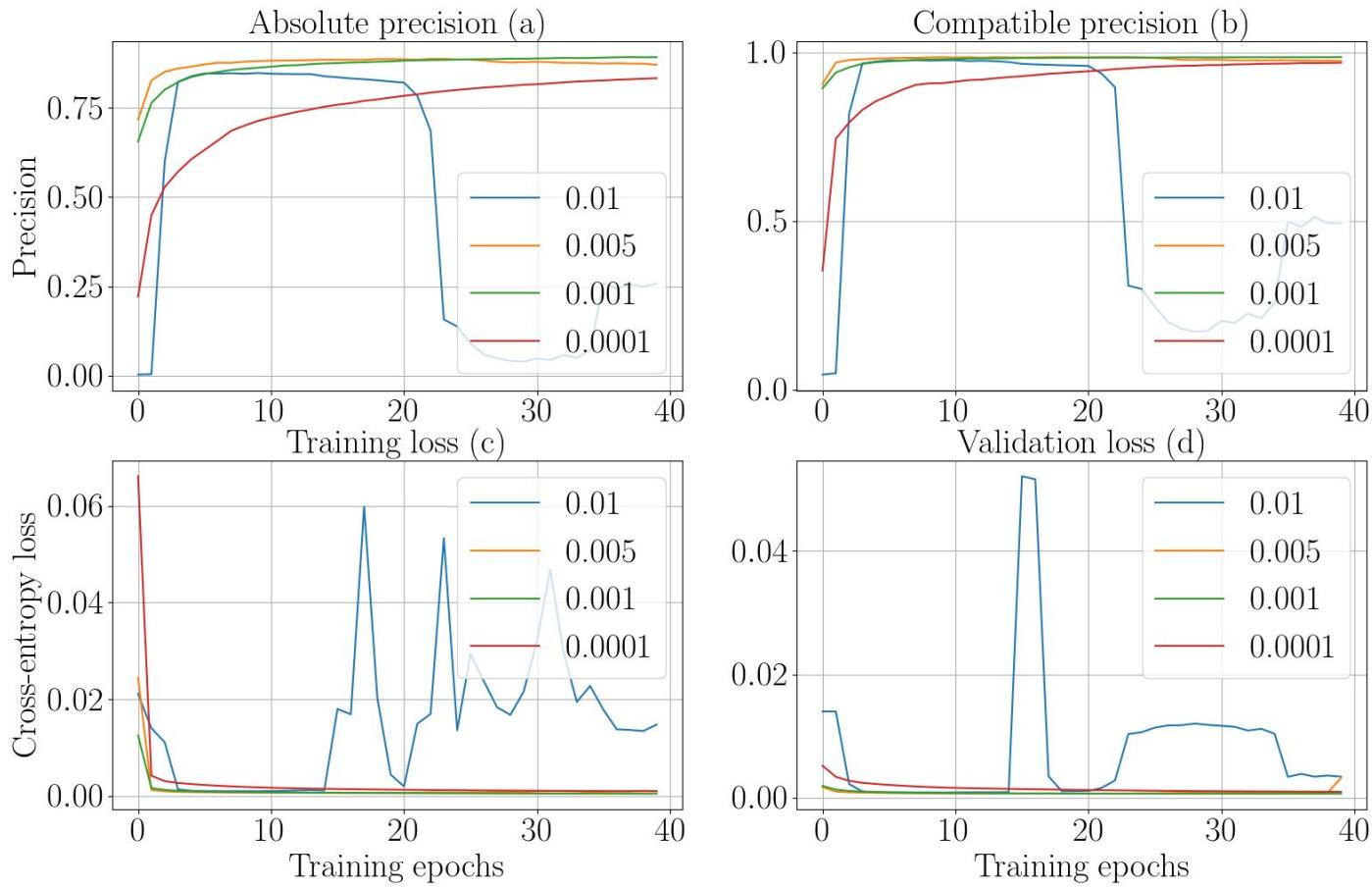
Distribution of number of compatible next tools



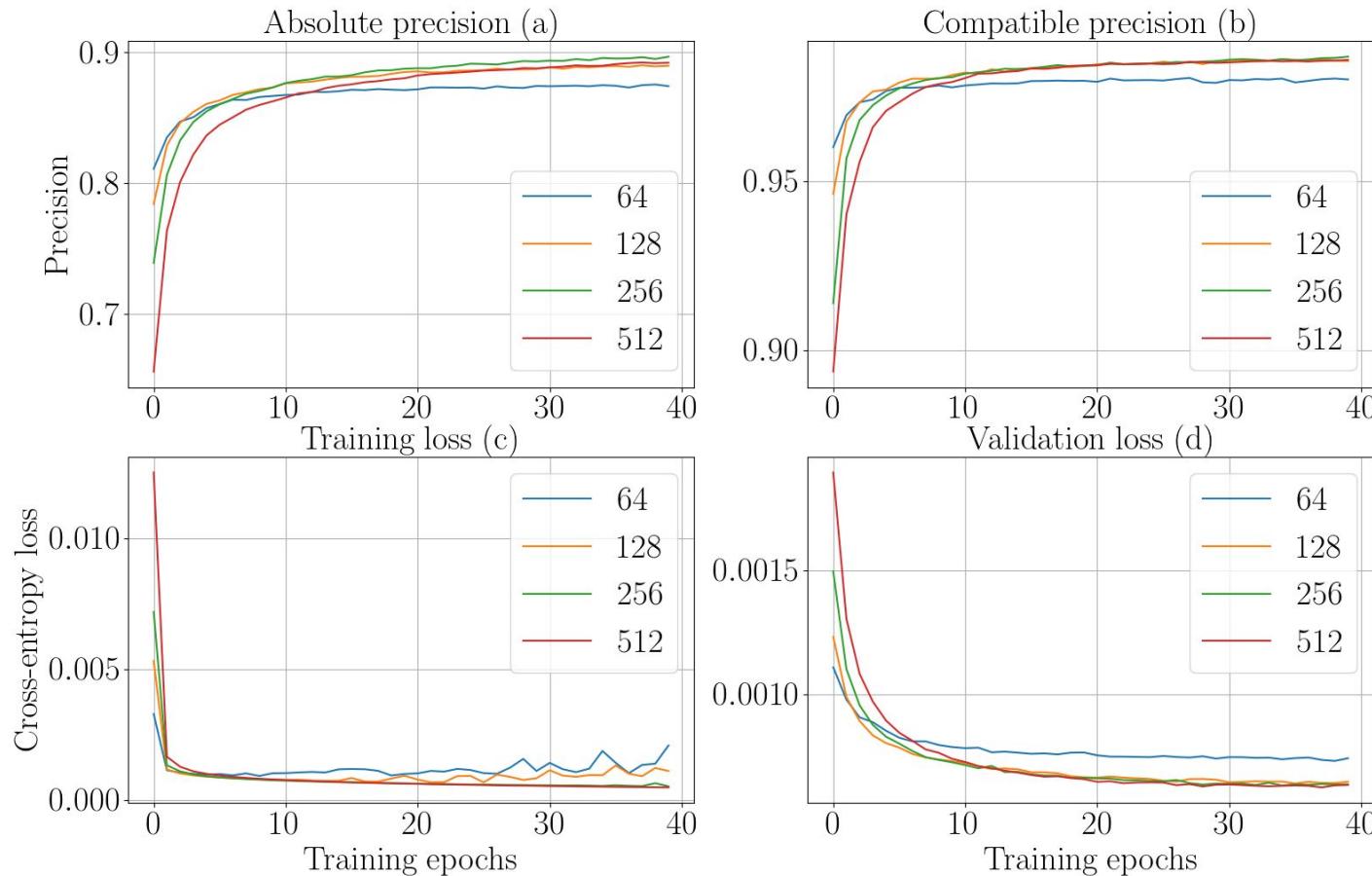
Precision and loss for various activations



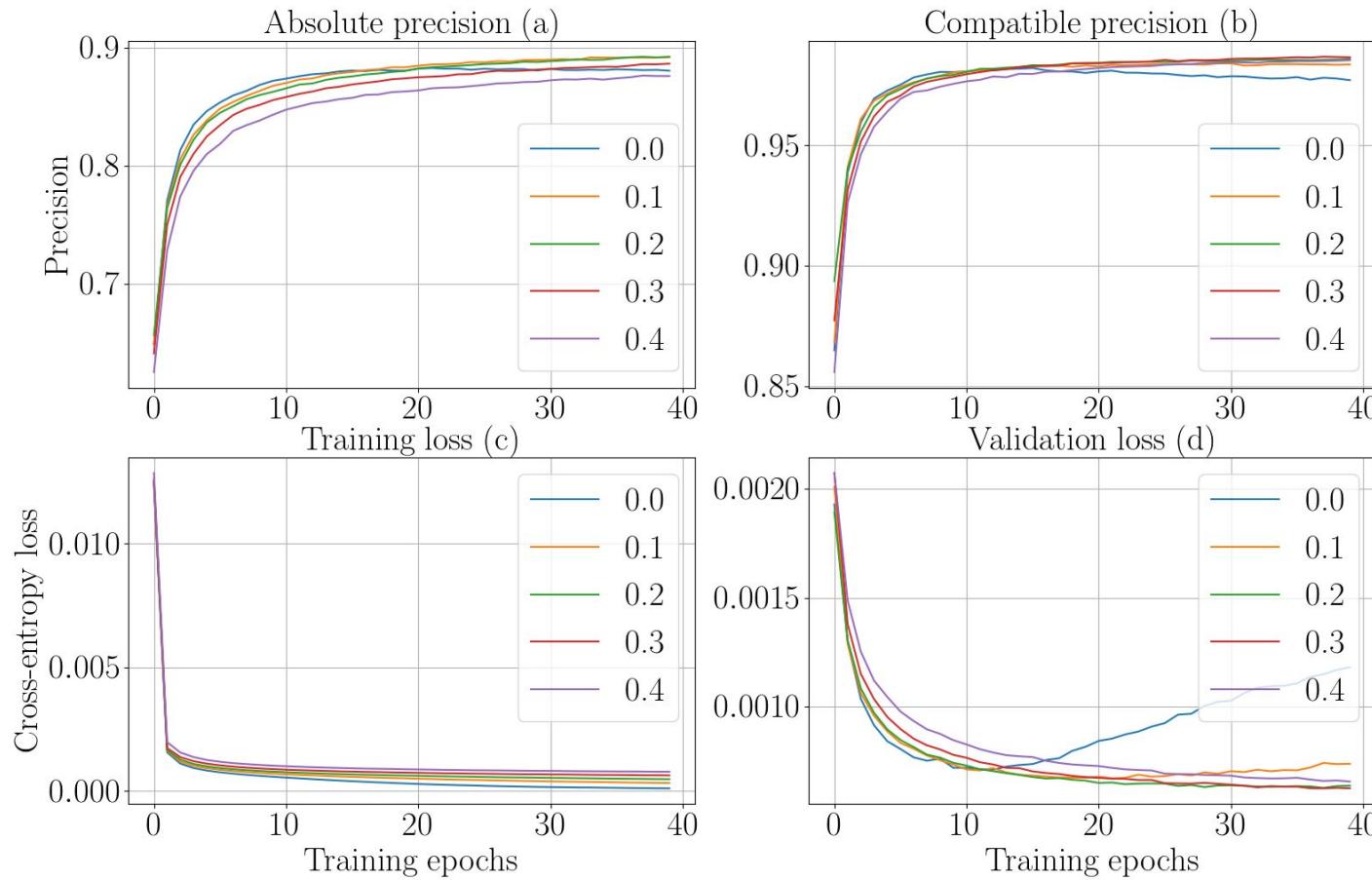
Precision and loss for various learning rates



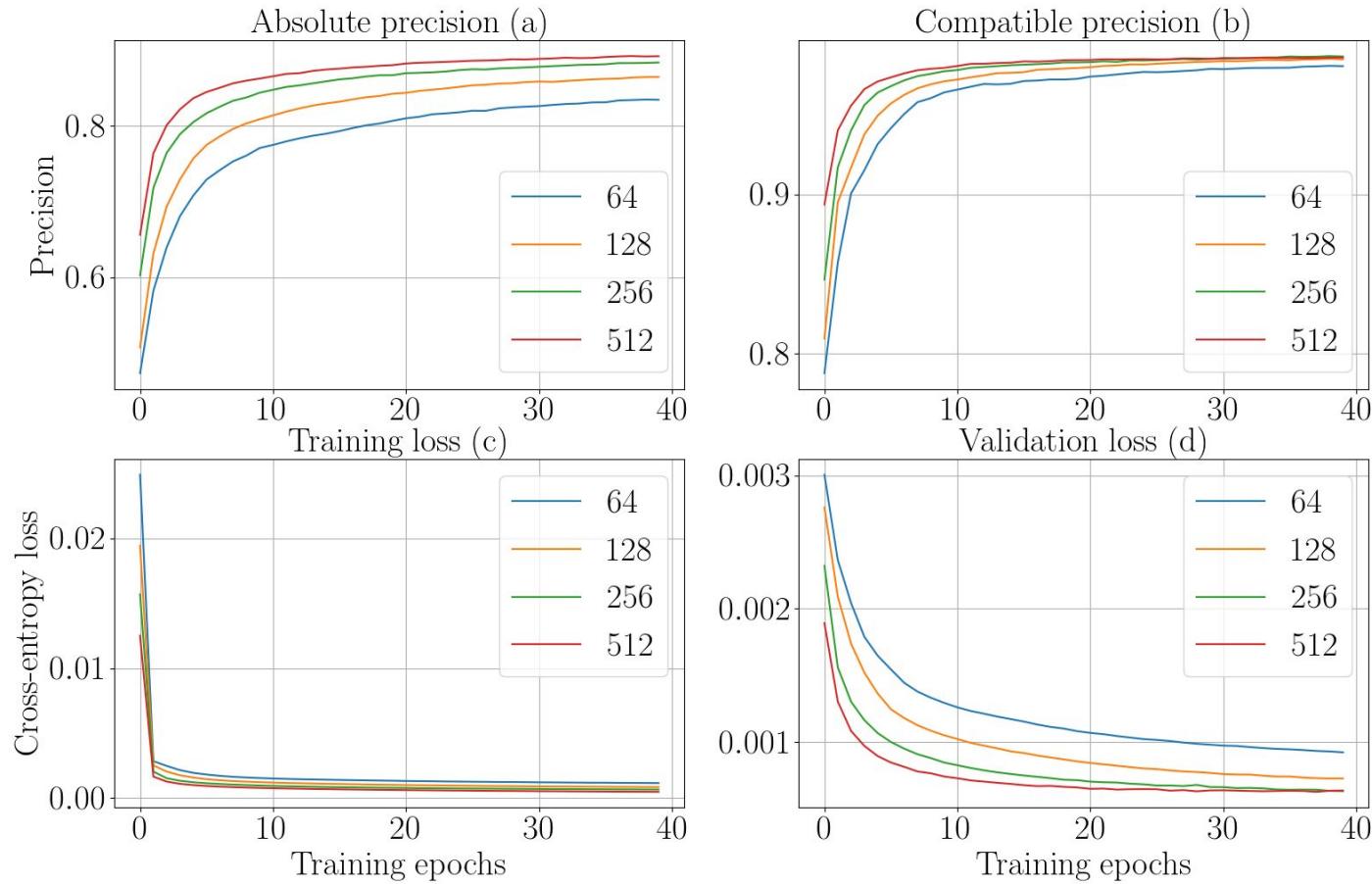
Precision and loss for various batch sizes



Precision and loss for various dropout values



Precision and loss for various number of memory units



Variation of precision with the length of paths

