

Recommendation system for scientific tools and workflows

Master's thesis

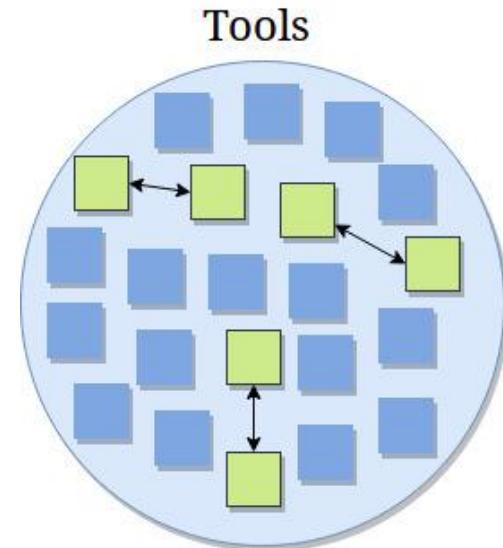
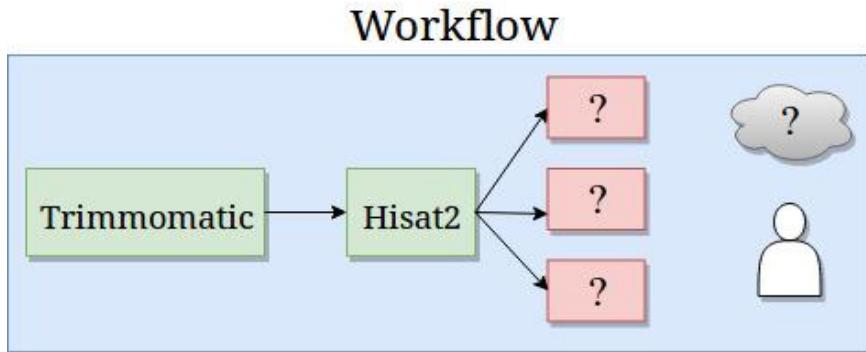
07/08/2018

Anup Kumar

Adviser: Dr. Björn Grüning

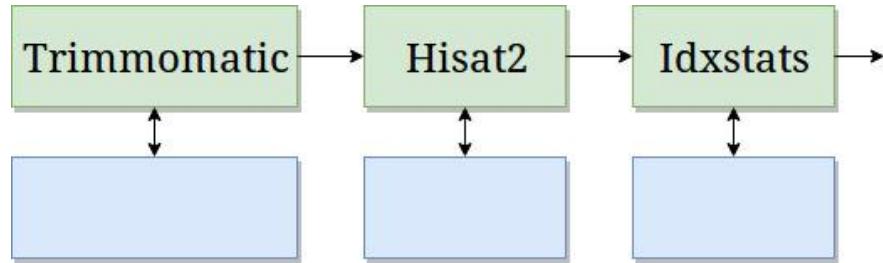
Motivation

- Galaxy - biological data analysis
 - Tools and workflows
 - Large number of tools (> 1,000)
 - Complex workflows
 - Need recommendation system

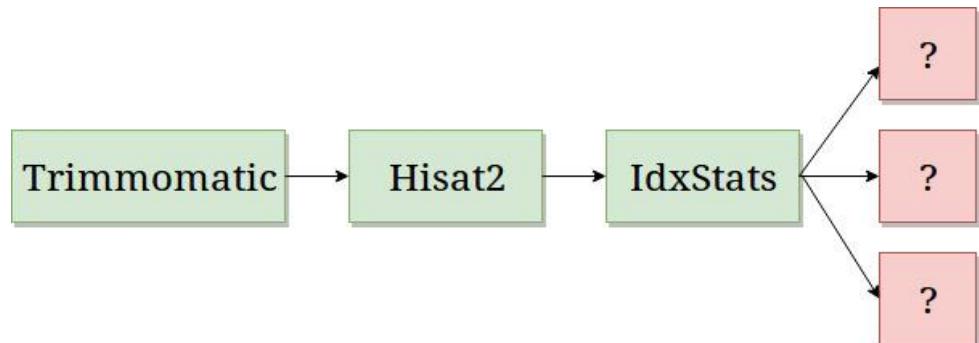


Recommendation system

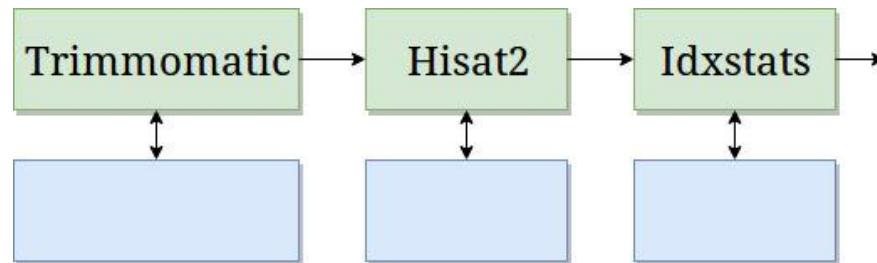
1. Find similar scientific tools



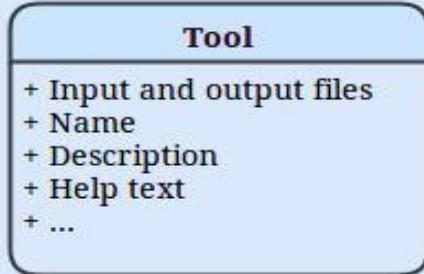
2. Predict tools in workflows



1. Find similar scientific tools



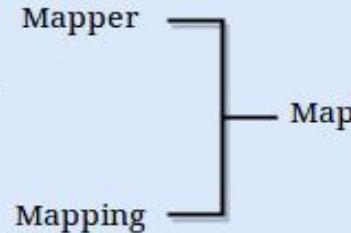
1. Read XML tools files



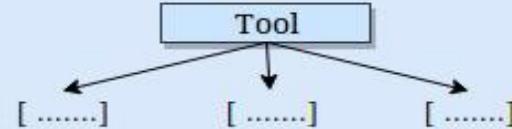
2. Clean metadata

- Stopwords: "at, in, for ..."

- Stemming



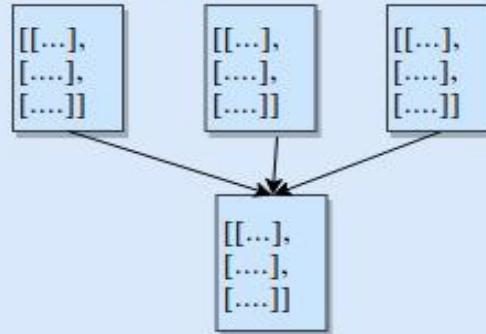
3. Vectors for tools

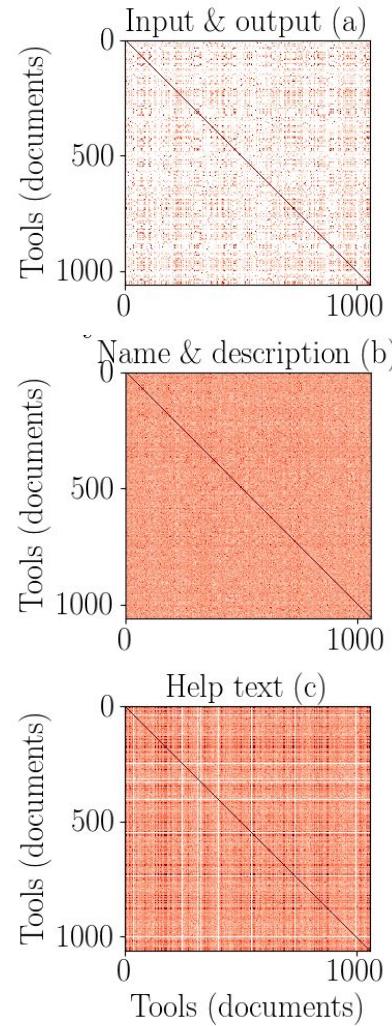


4. Similarity between vectors



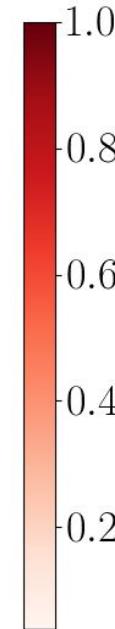
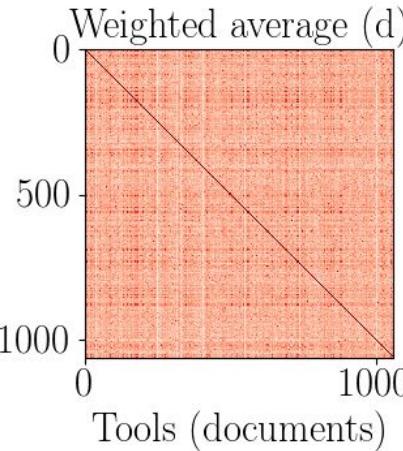
5. Combine similarity matrices





Combine similarity matrices

Tools (documents)



Optimisation

- Gradient descent
- Learn weights on similarity scores
- Mean squared error

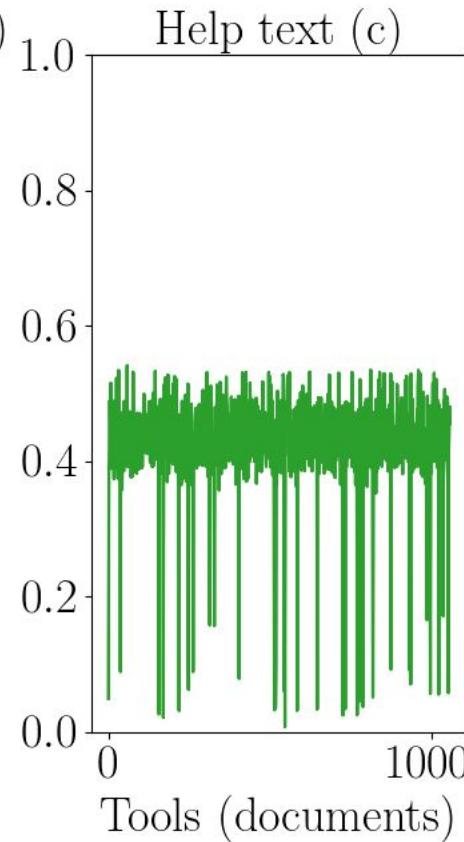
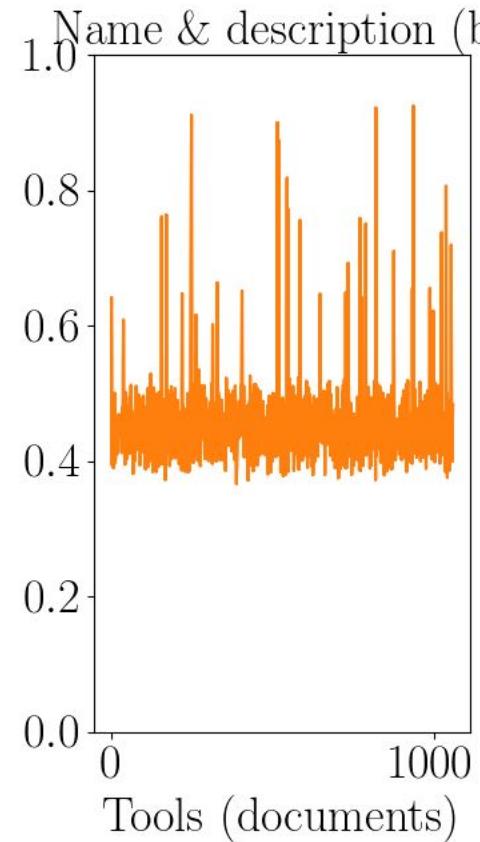
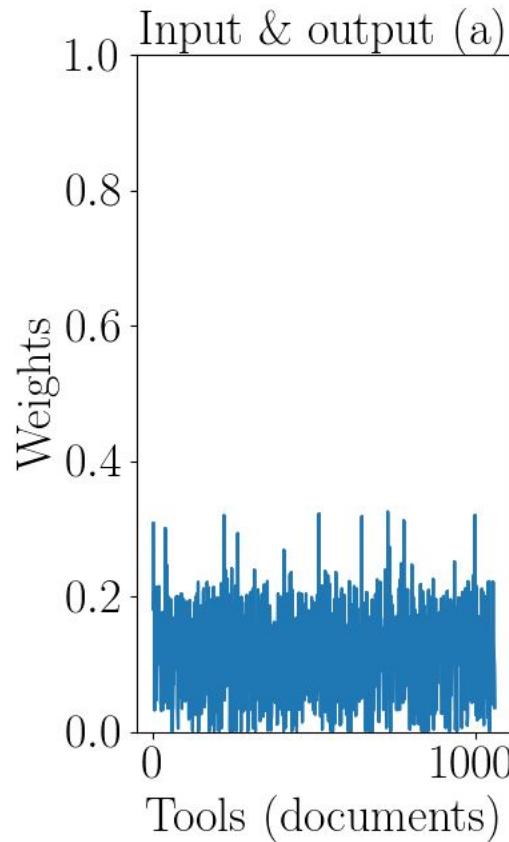
$$Error(w^k) = \frac{1}{N} \times \sum_{j=1}^N [w^k \times SM^k - SM_{ideal})^2]_j$$

- Each tool, three weights
- Obtain a weighted average similarity matrix

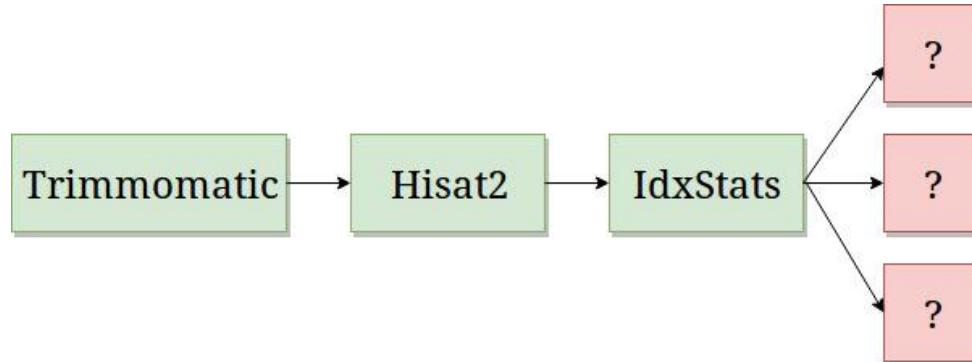
True similarity is an array of 1.0

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

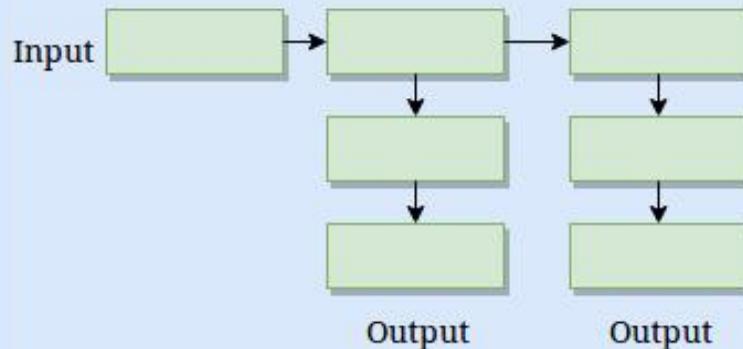
Distribution of weights



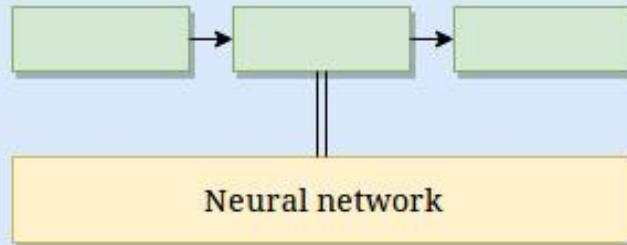
2. Predict tools in workflows



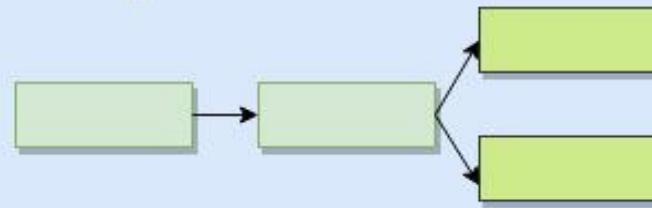
1. Extract paths from workflows



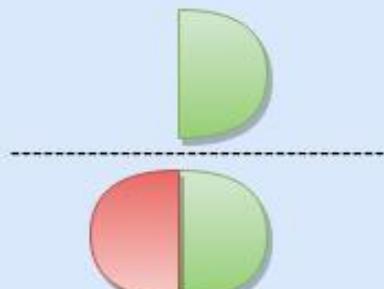
2. Preprocess the paths



3. Classify and predict tools using a neural network



4. Compute precision



Workflows

- Workflow - a directed acyclic graph
- ~ 193,000 workflows
- 900,000 paths (~ 167,000 unique)
- Maximum 25 tools in a path

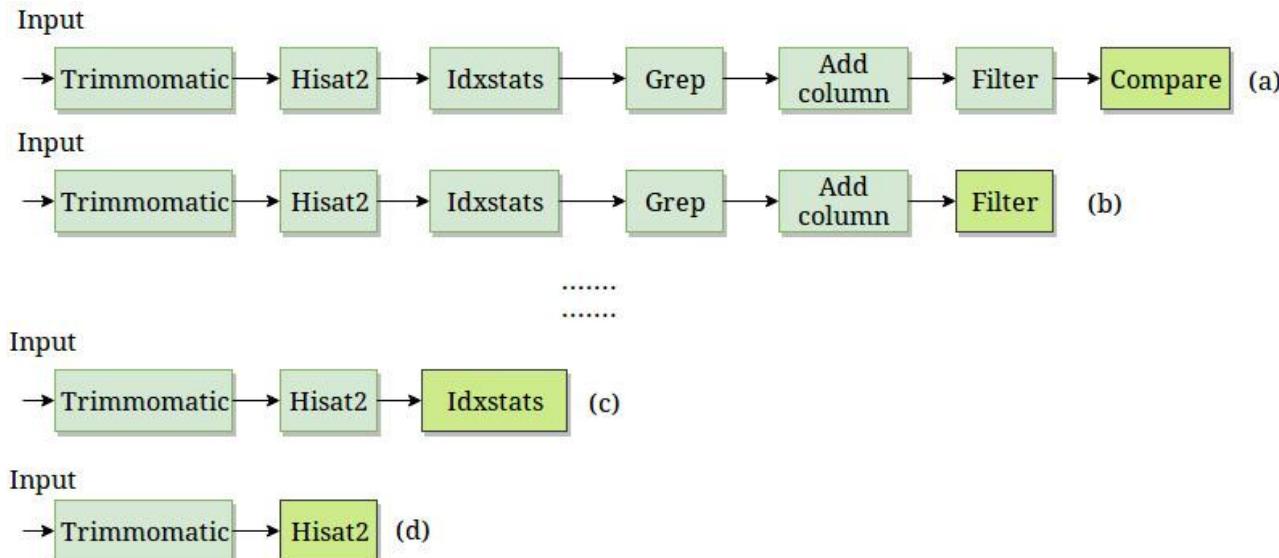
Directed acyclic graph:

<https://cran.r-project.org/web/packages/ggdag/vignettes/intro-to-dags.html>

<https://galaxyproject.org/learn/advanced-workflow/>

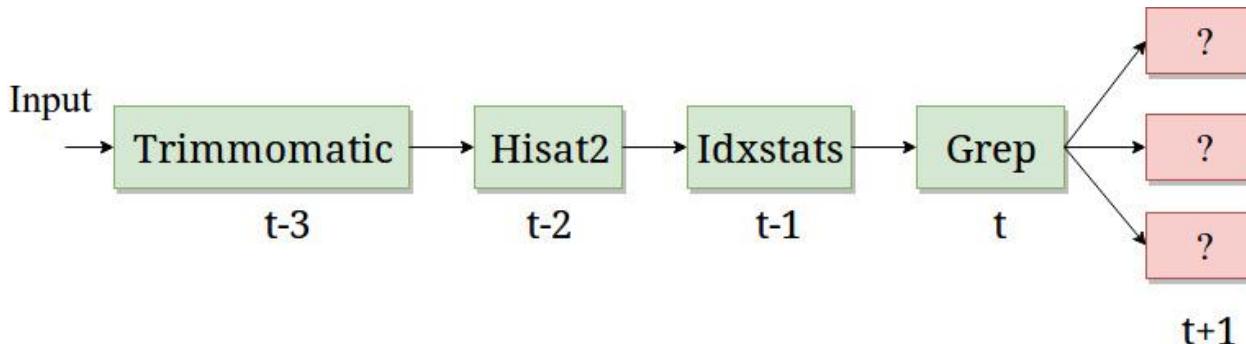
Workflow preprocessing

- Decomposition
- Last tool is a label



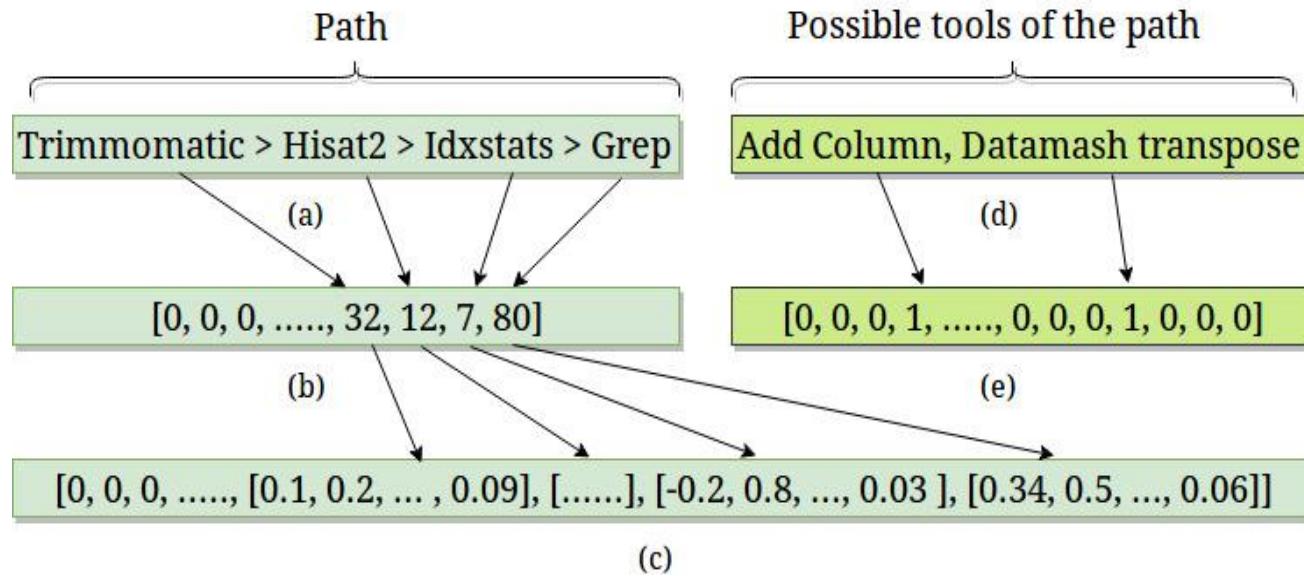
Recurrent neural network

- 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>

Embedding and label vectors



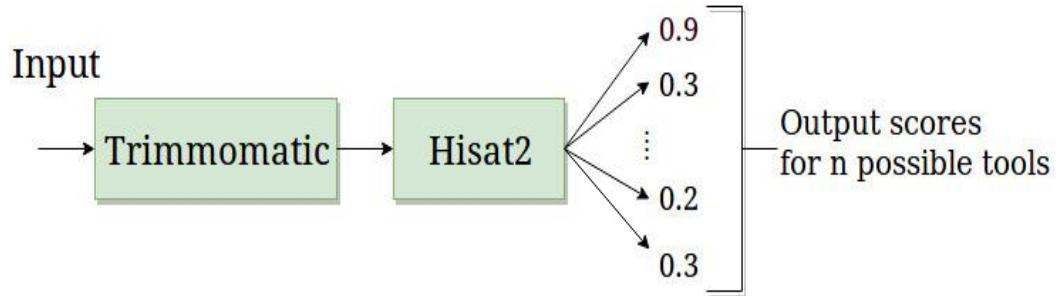
Dimensions:

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

Prediction and precision

- Score for each predicted tool
- Sigmoid activation

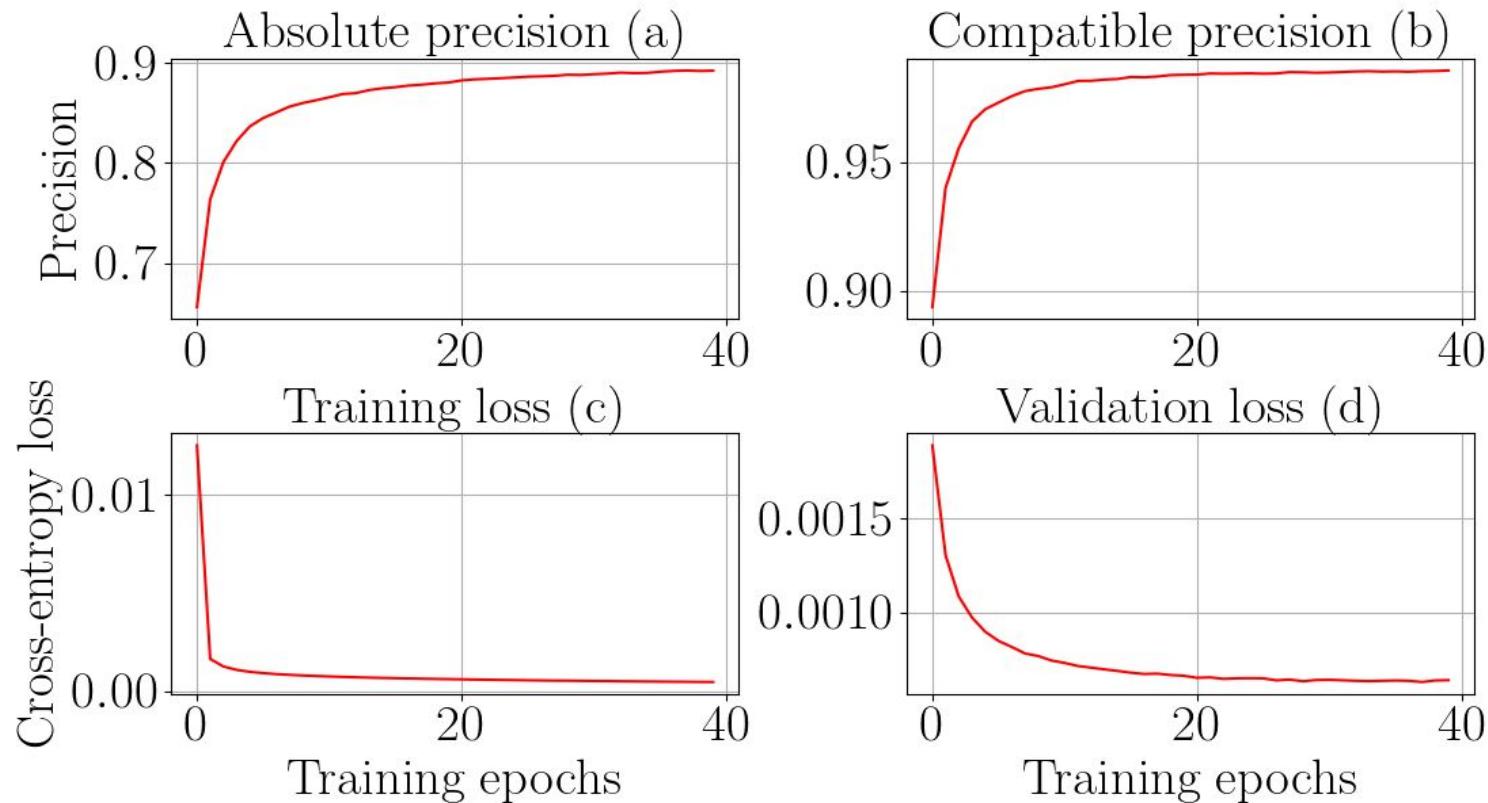
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



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

Sigmoid: <https://cs231n.github.io/neural-networks-1/>

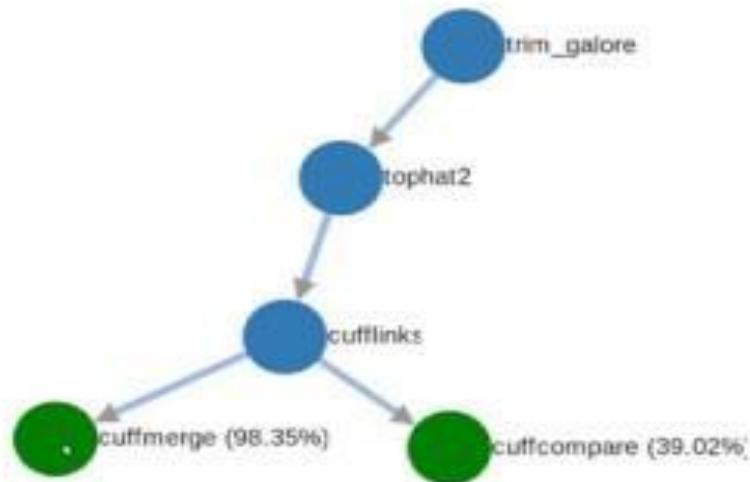
Precision and loss for decomposition of train and test paths



Predict Next Tools For Galaxy Workflows

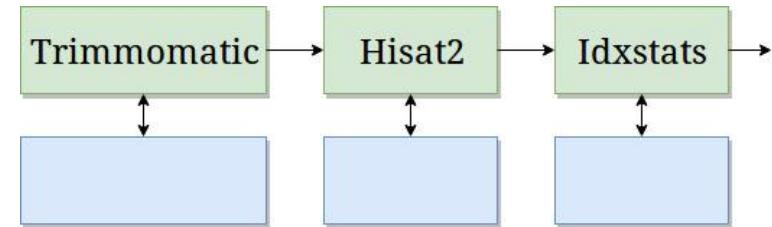
Visit to see workflows as Cytoscape graphs

Choose a tool



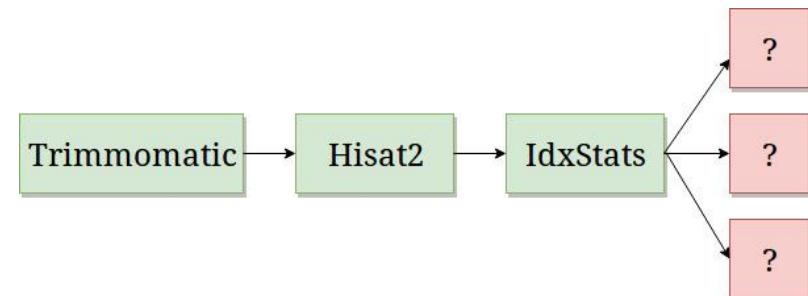
Conclusion and future work (part 1)

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



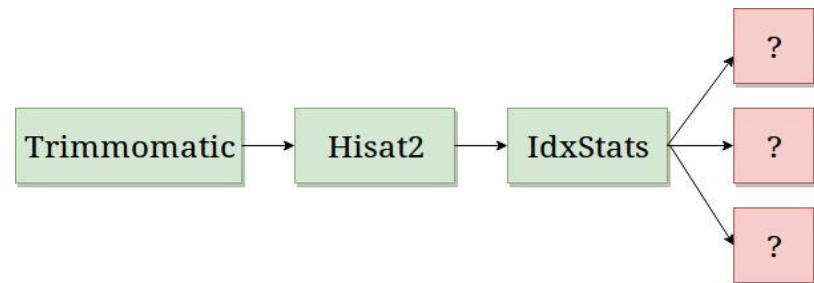
Summary and conclusion (part 2)

- Workflows - directed acyclic graphs (193,000)
- Extract paths (167,000 unique) from workflows
- Recurrent neural network (gated recurrent units)
- Multilabel, multiclass classification
- Absolute precision ~ 89%, compatible precision ~ 99% (~ 48 hrs)
- More workflows, better precision
- Recommendation system using similar and predicted tools



Future work (part 2)

- Restore original distribution
- Decay prediction over time
- Oversample
- Integrate into Galaxy



Thank you all!

- Prof. Dr. Rolf Backofen
- Prof. Dr. Wolfgang Hess
- Dr. Björn Grüning
- Dr. Anika Erxleben
- Freiburg Galaxy team
- Nate Coraor (Galaxy team, Penn State University)

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

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

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

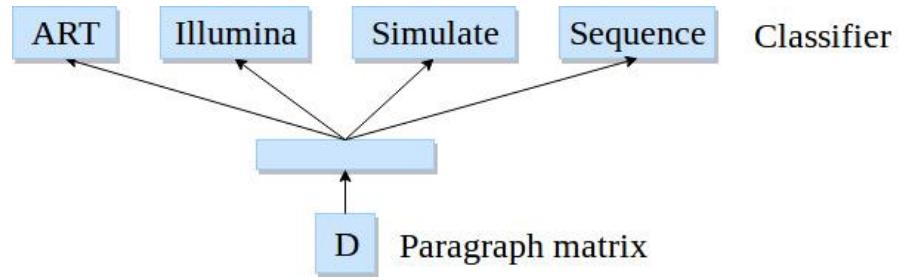
$$V^T \cdot V = I_{m \times m}$$

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

Paragraph vectors

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

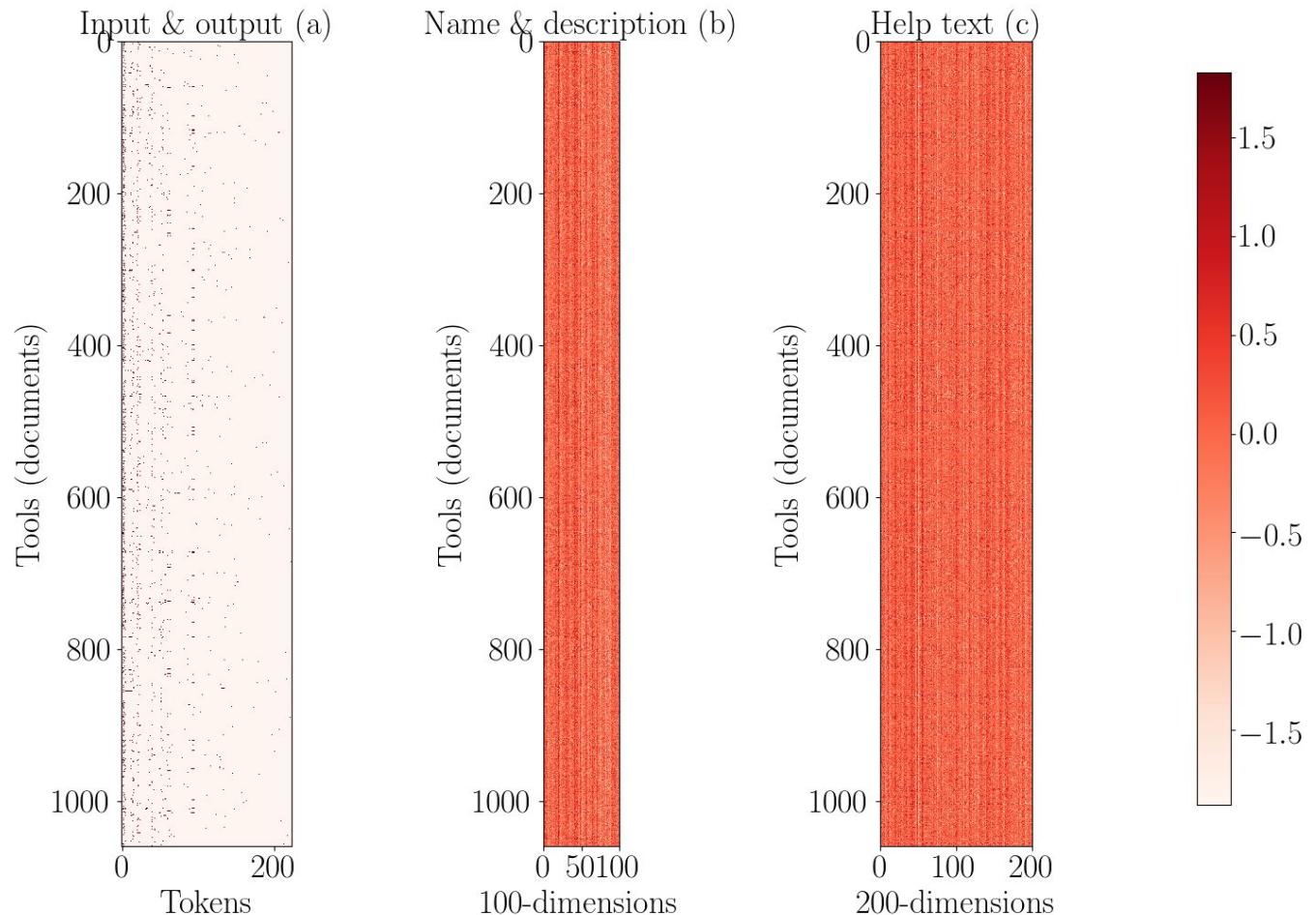
$$\frac{1}{T} \cdot \sum_{t=k}^{T-k} \log p(w_t | w_{t-k}, \dots, w_{t+k})$$



*<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

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

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

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

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

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

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

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

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

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

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

Optimisation

- Gradient descent
- Learn weights on similarity scores

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

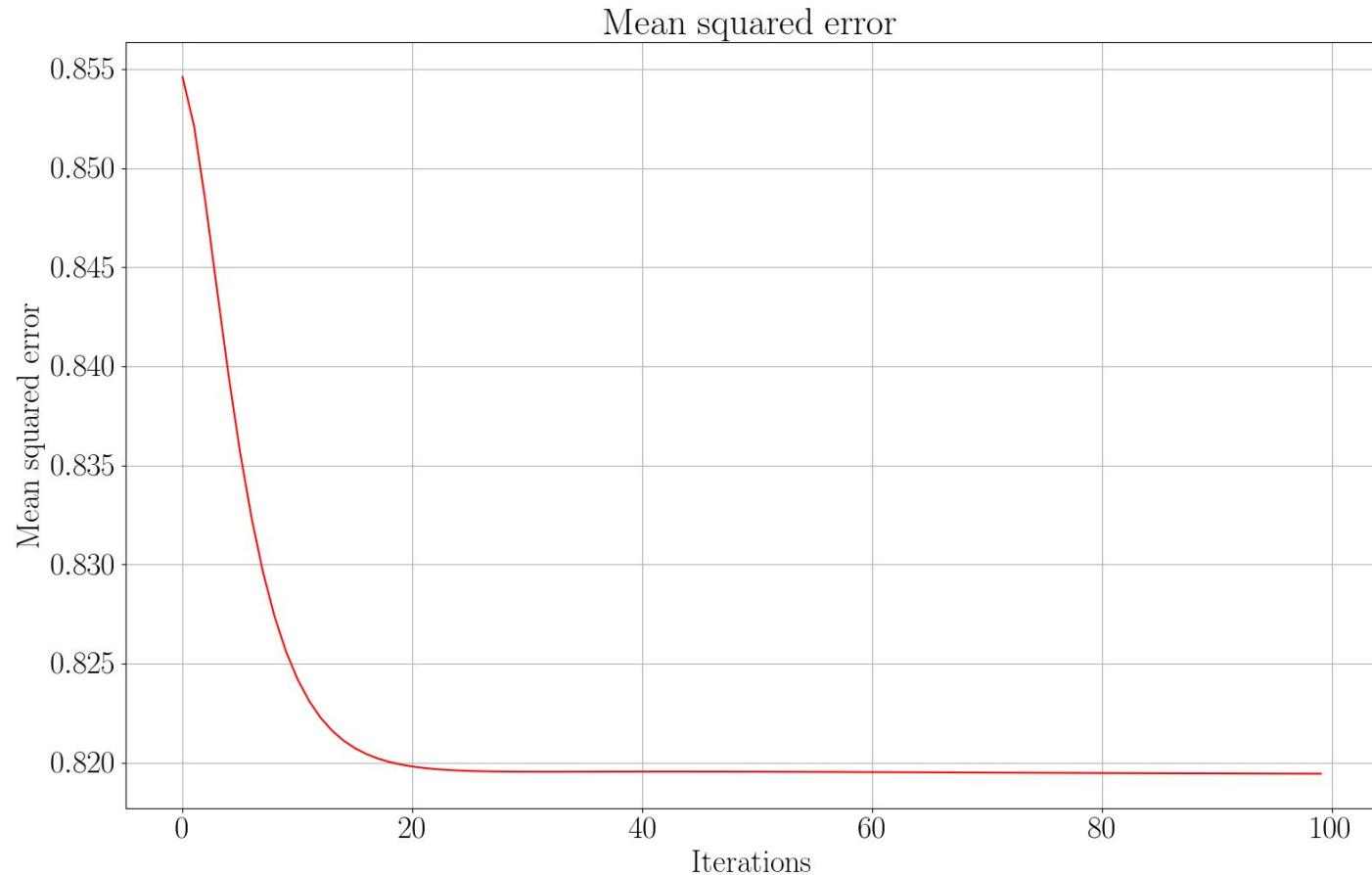
$$Gradient(w^k) = \frac{\partial Error}{\partial w^k} = \frac{2}{N} \times ((w^k \times SM^k - SM_{ideal}) \cdot SM^k) \quad \dots (2)$$

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

- Obtain a weighted average similarity matrix

SM_{ideal} is an array of 1.0

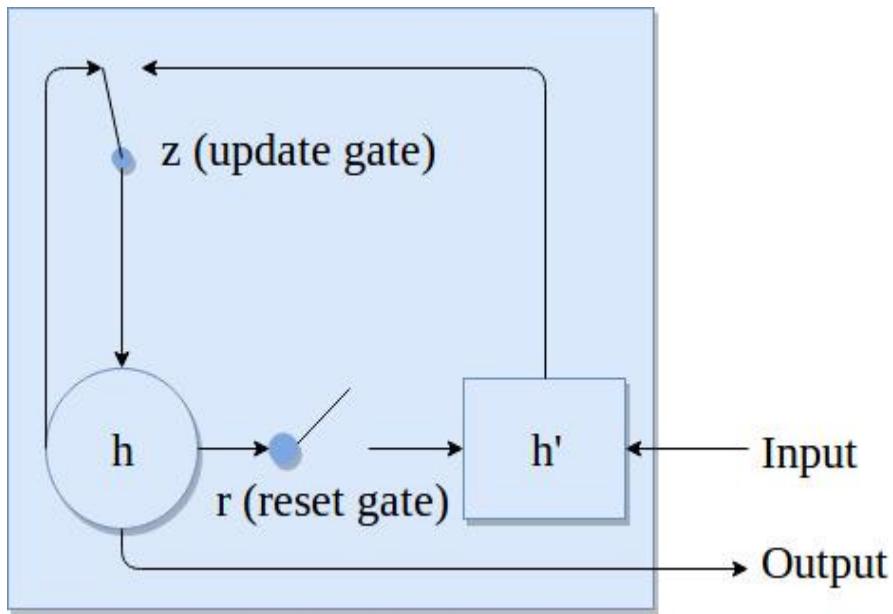
Gradient descent: <https://developers.google.com/machine-learning/crash-course/reducing-loss/gradient-descent>



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:

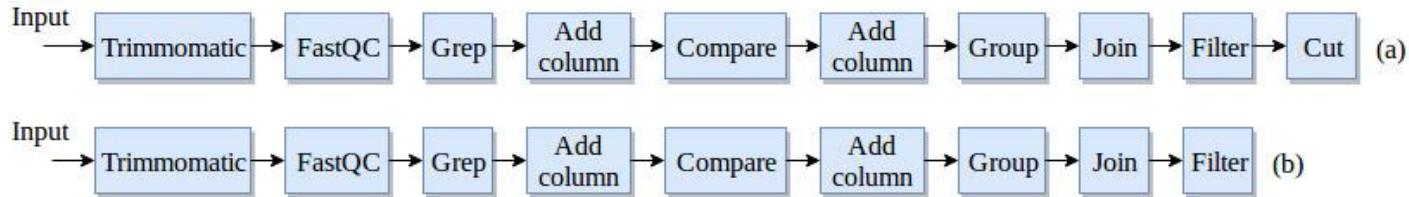
Total paths (111,386), train paths (89,109) and test paths (22,277).

- Decomposition:

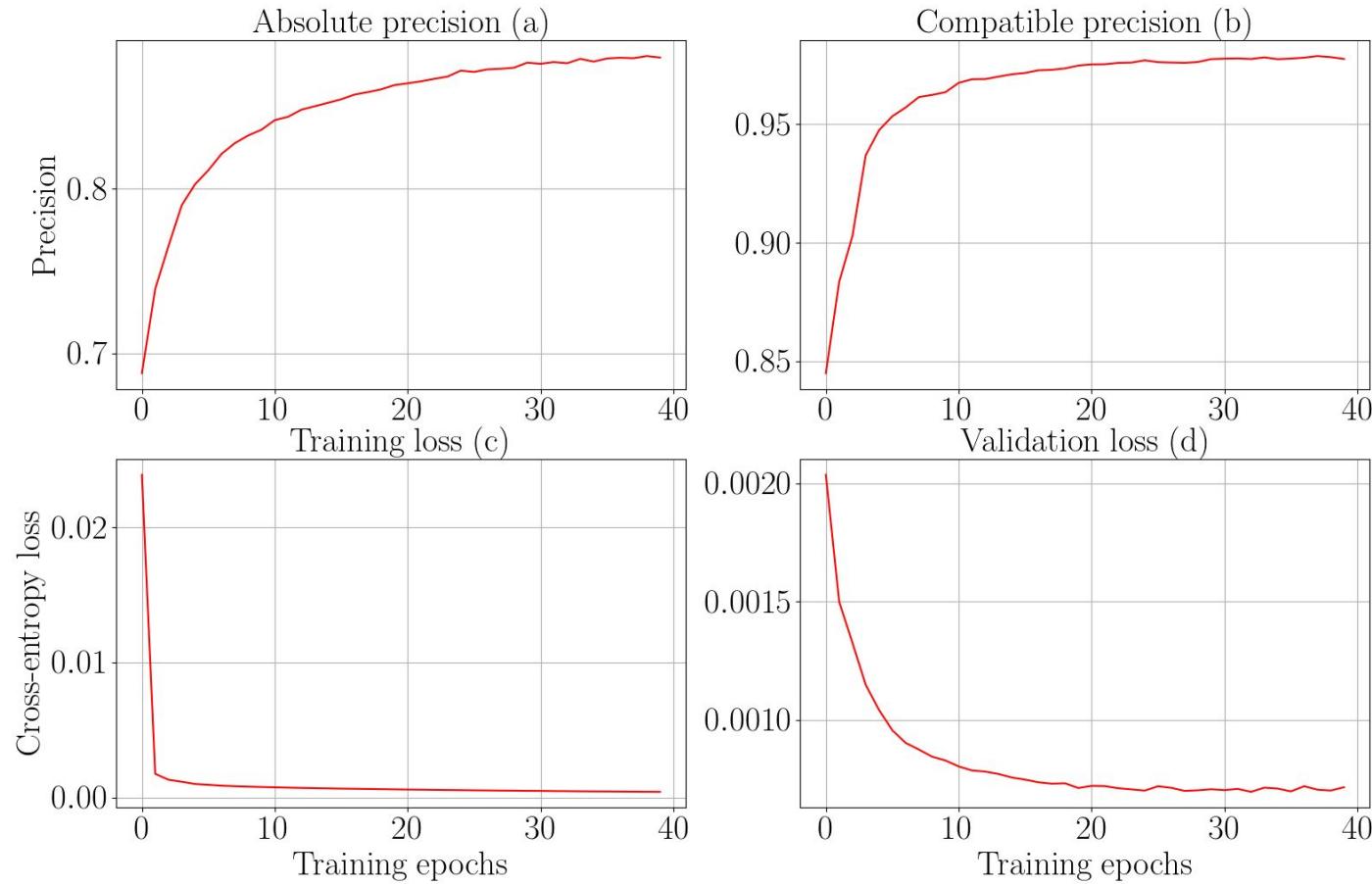
Total paths (210,983), train paths (168,787) and test paths (42,196).

Workflow preprocessing

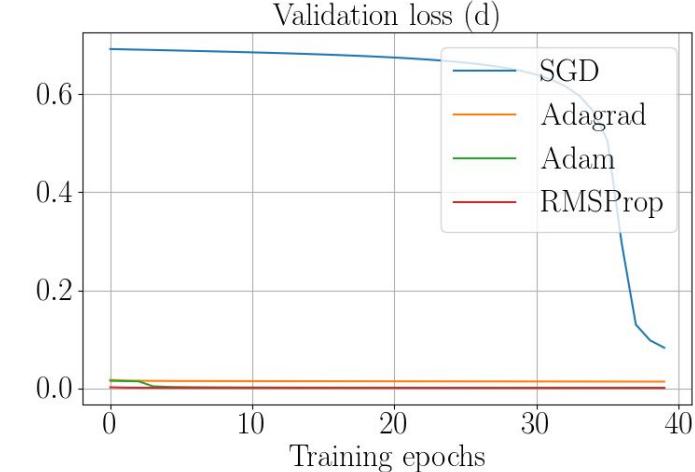
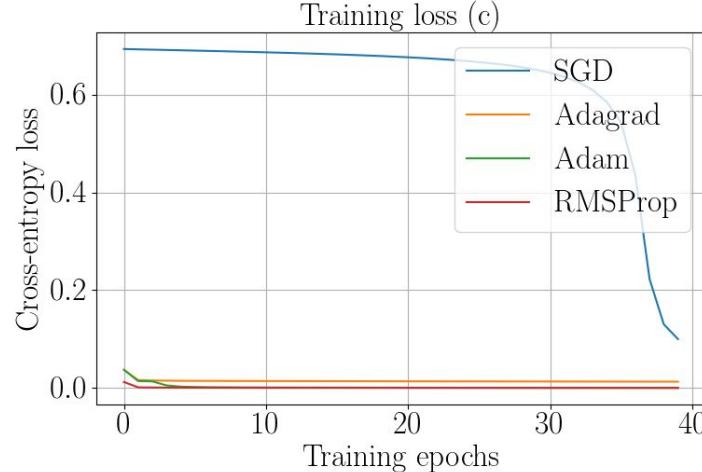
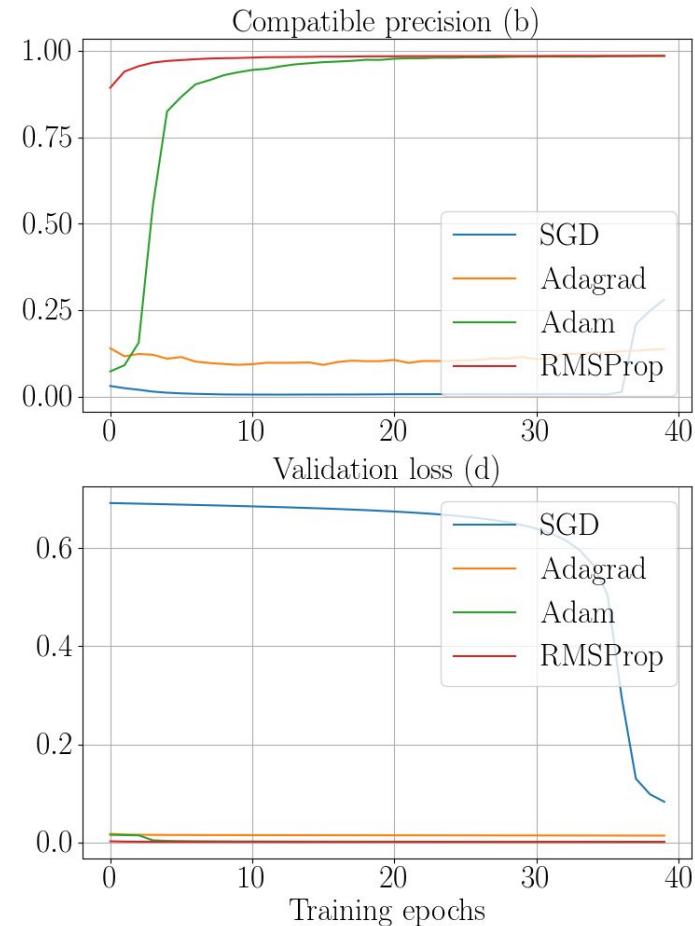
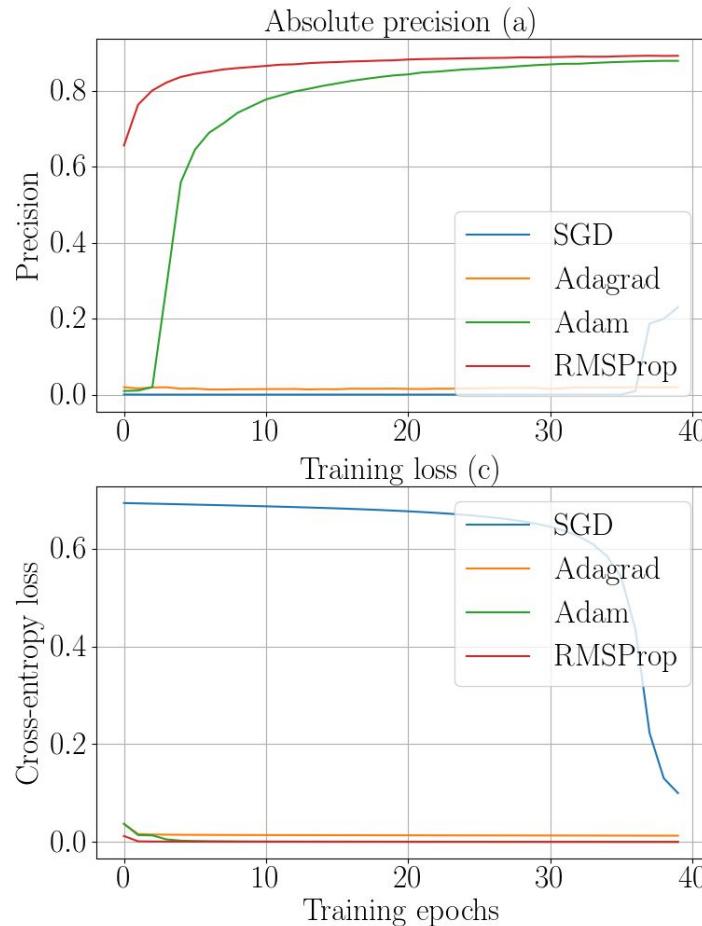
- No decomposition
- Last tool is a label



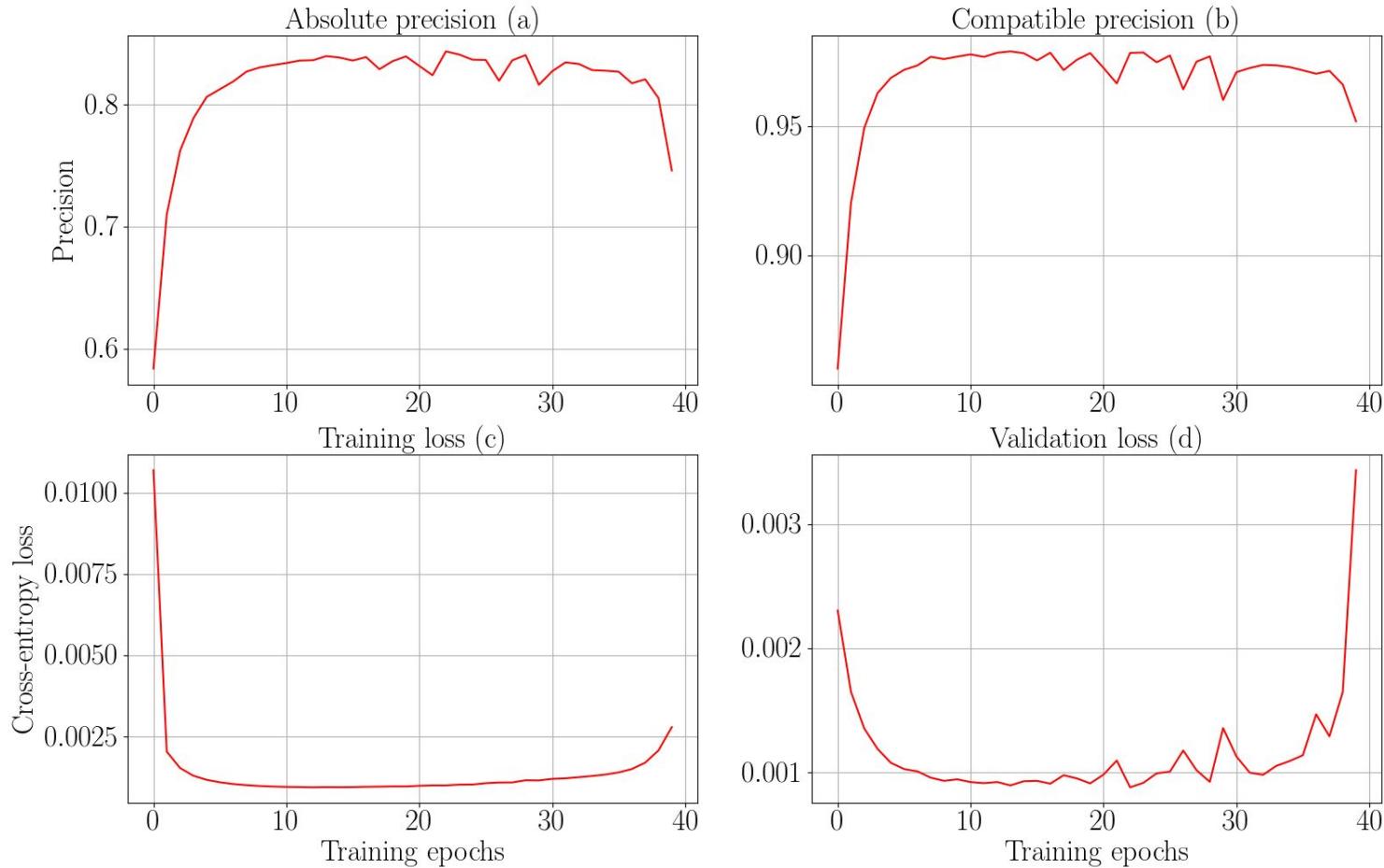
Precision and loss for no decomposition of paths



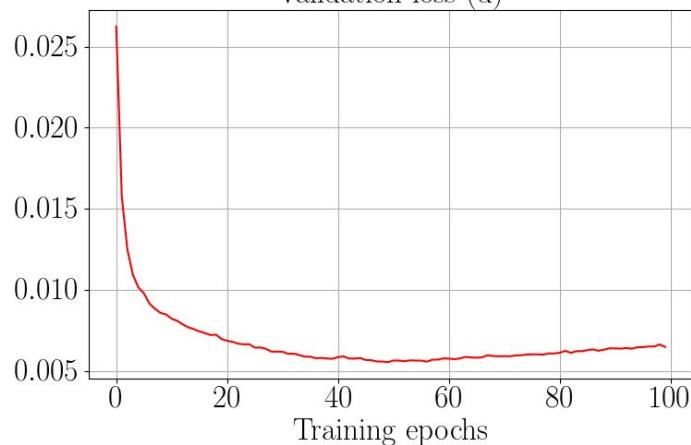
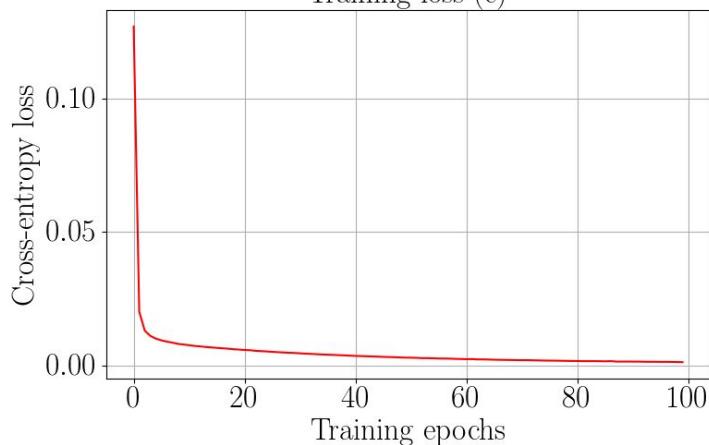
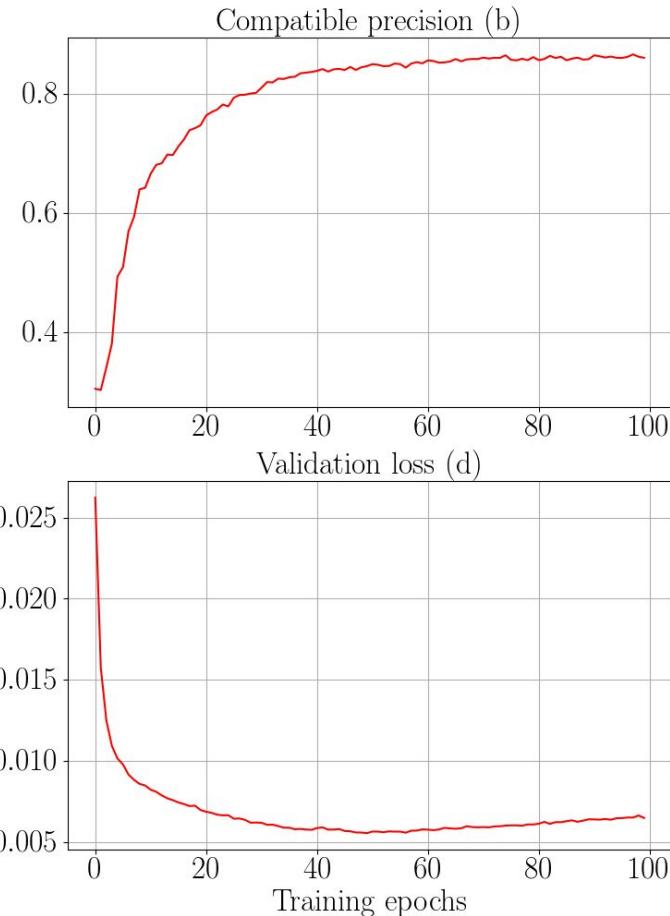
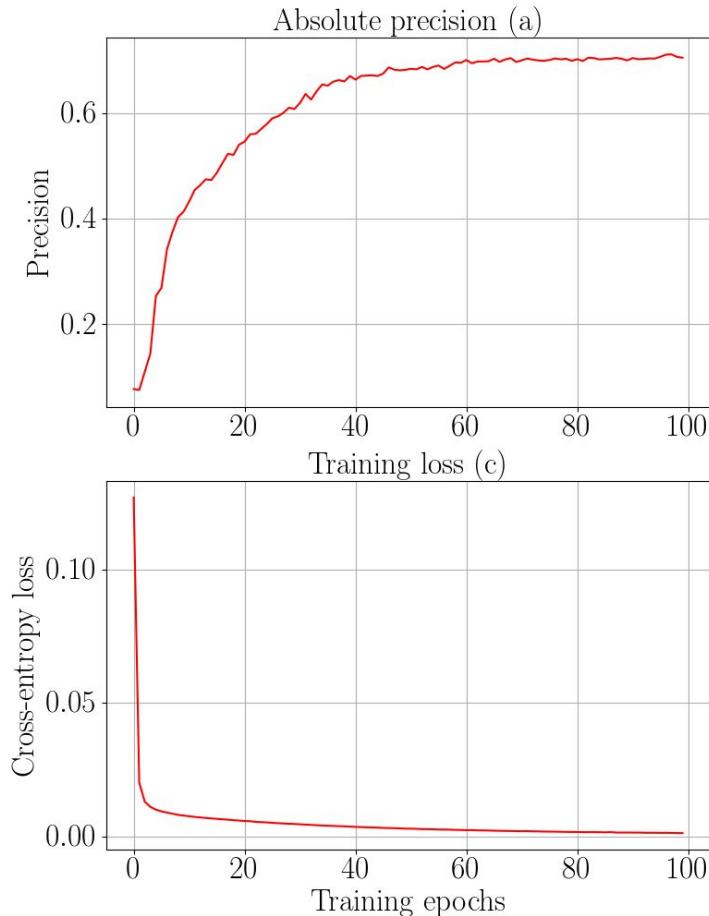
Precision and loss for various optimisers



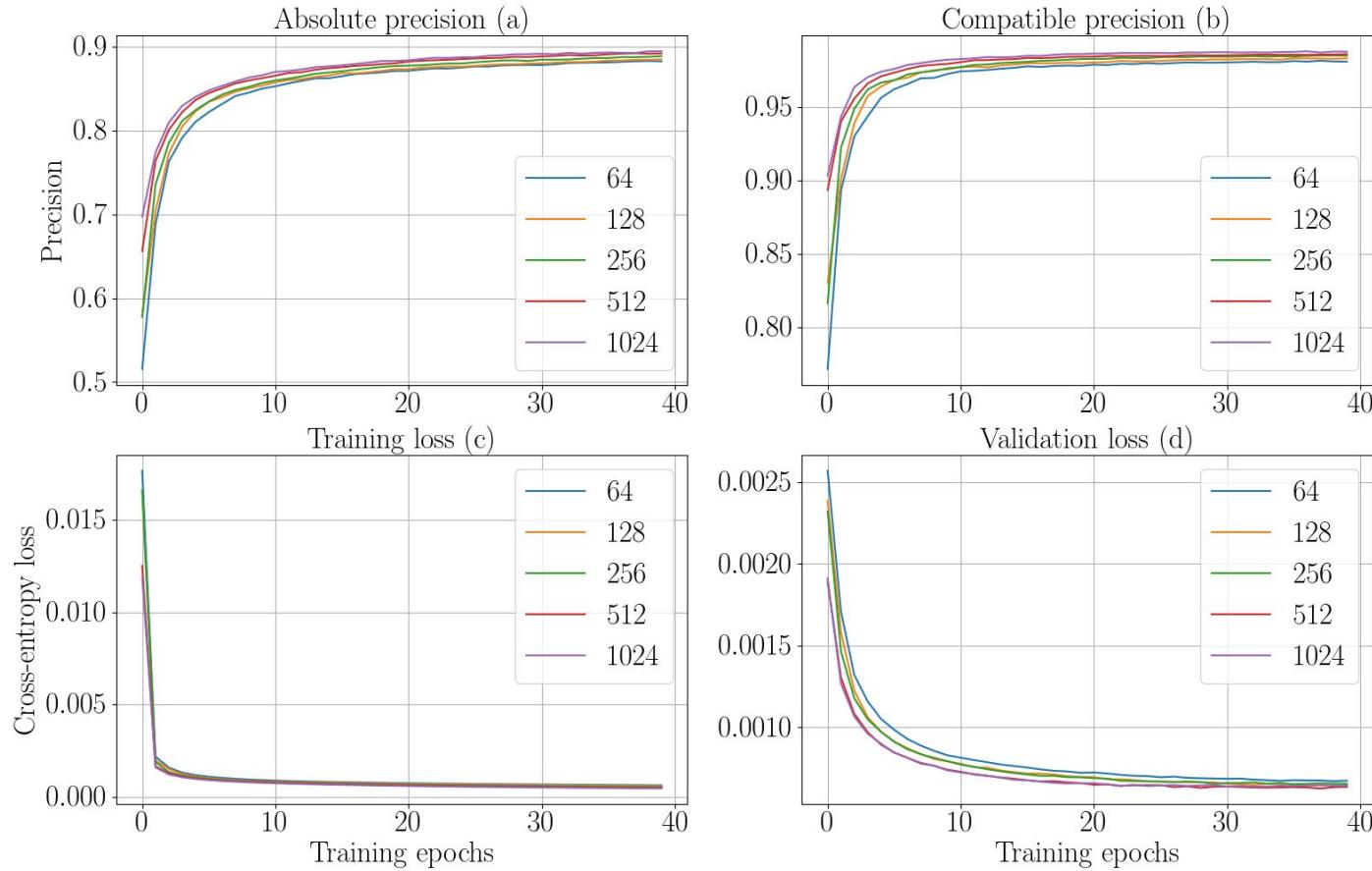
Precision and loss using neural network with dense layers



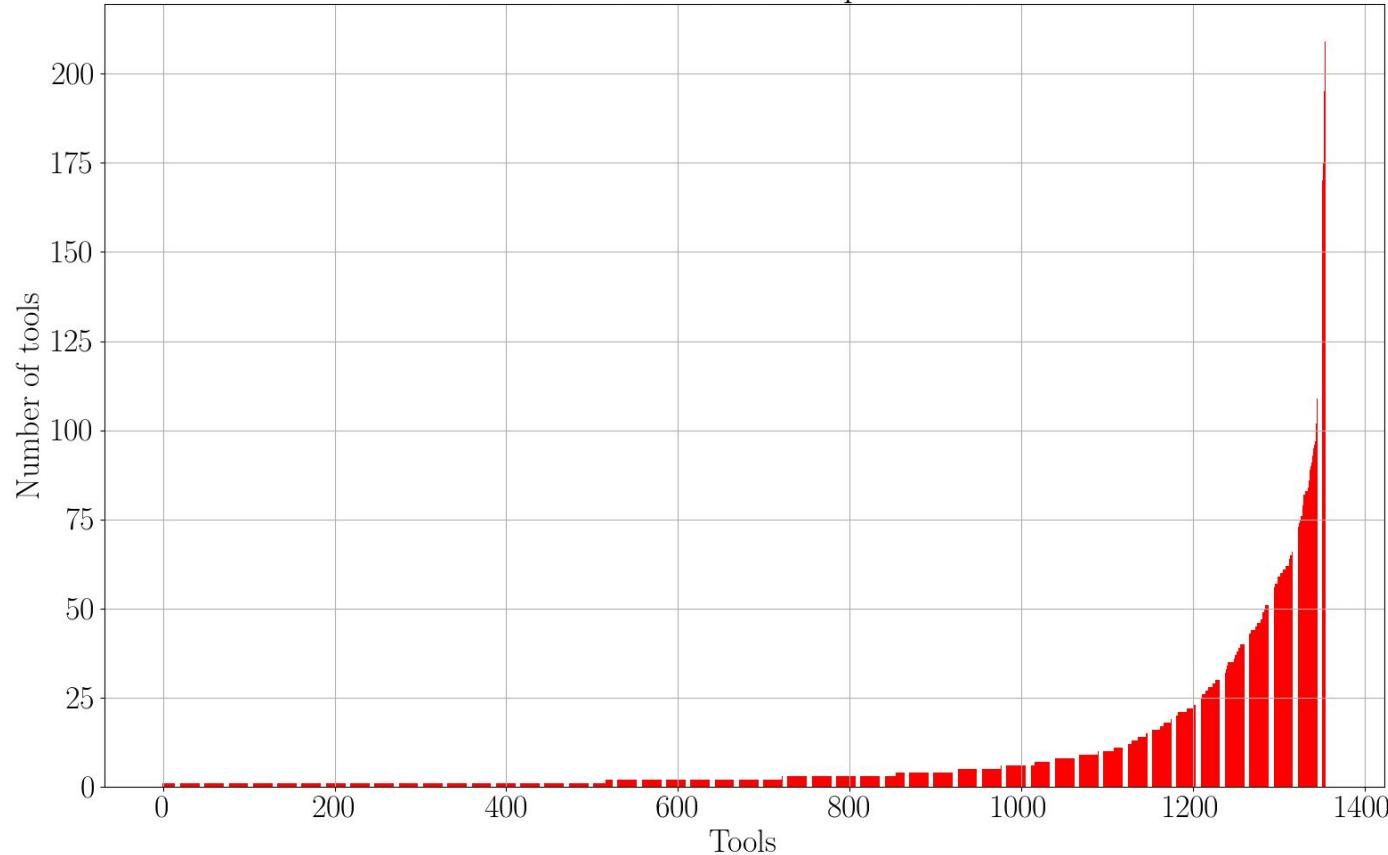
Precision and loss using less data



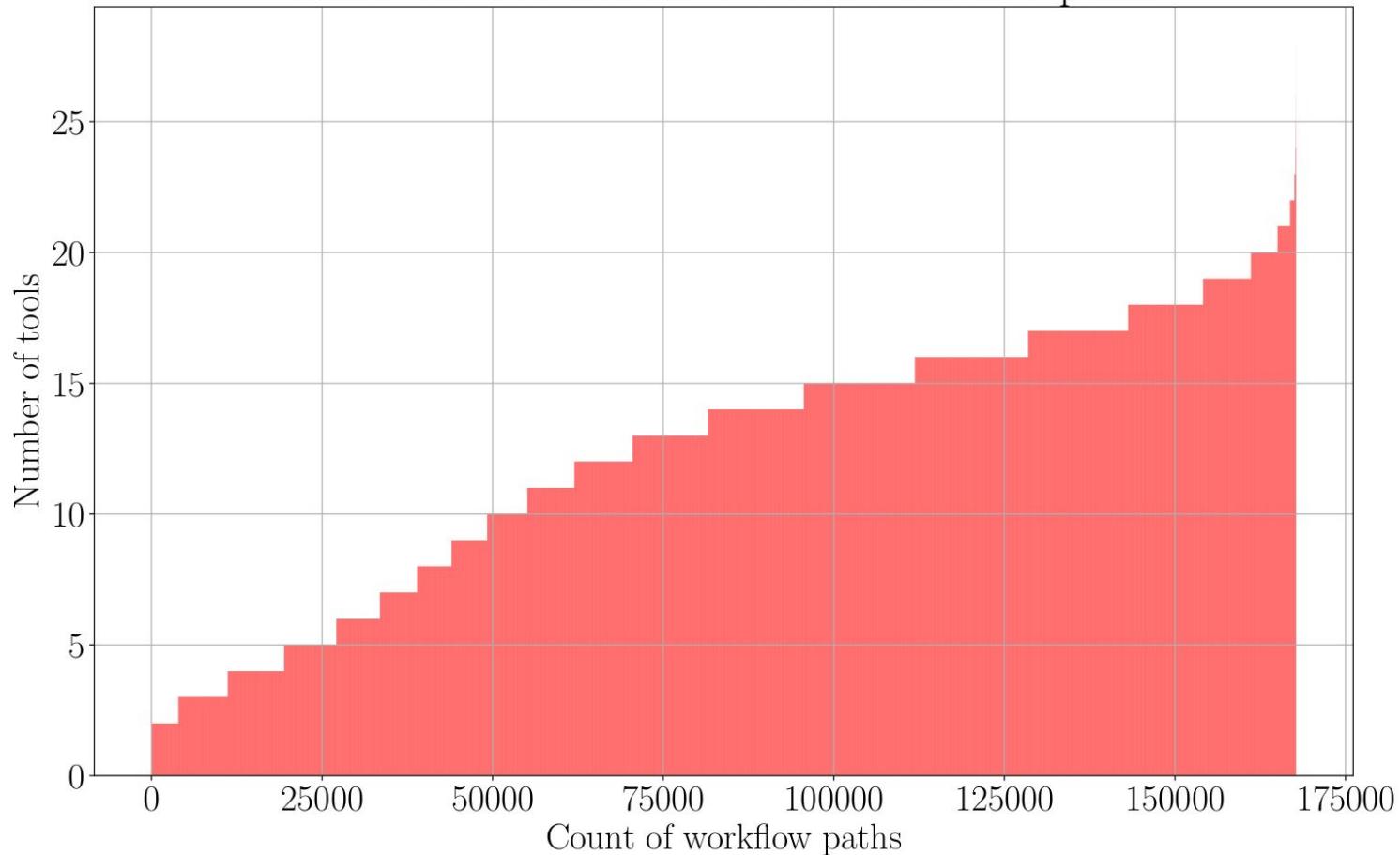
Precision and loss for various sizes of embedding layer



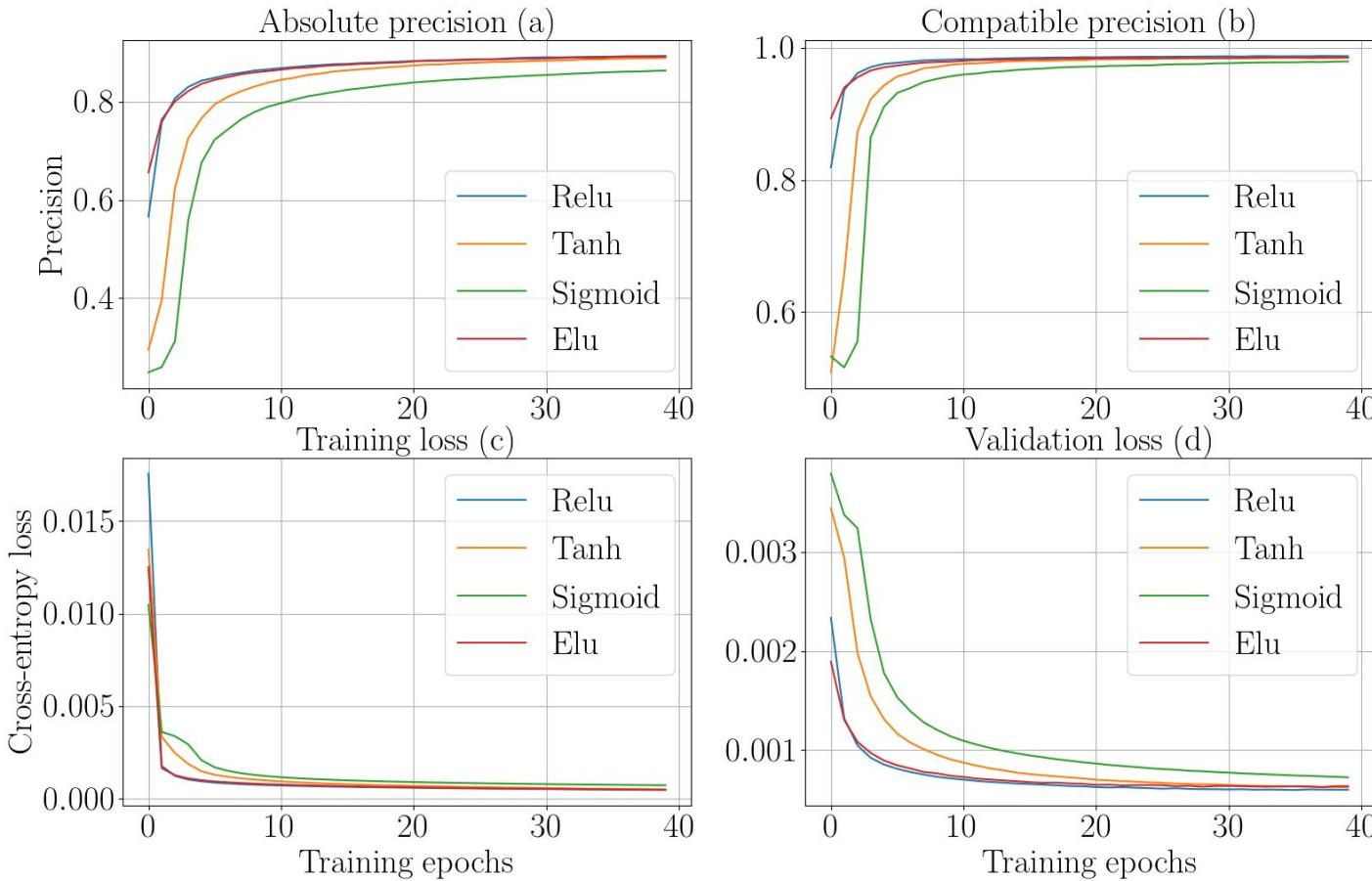
Distribution of number of compatible next tools



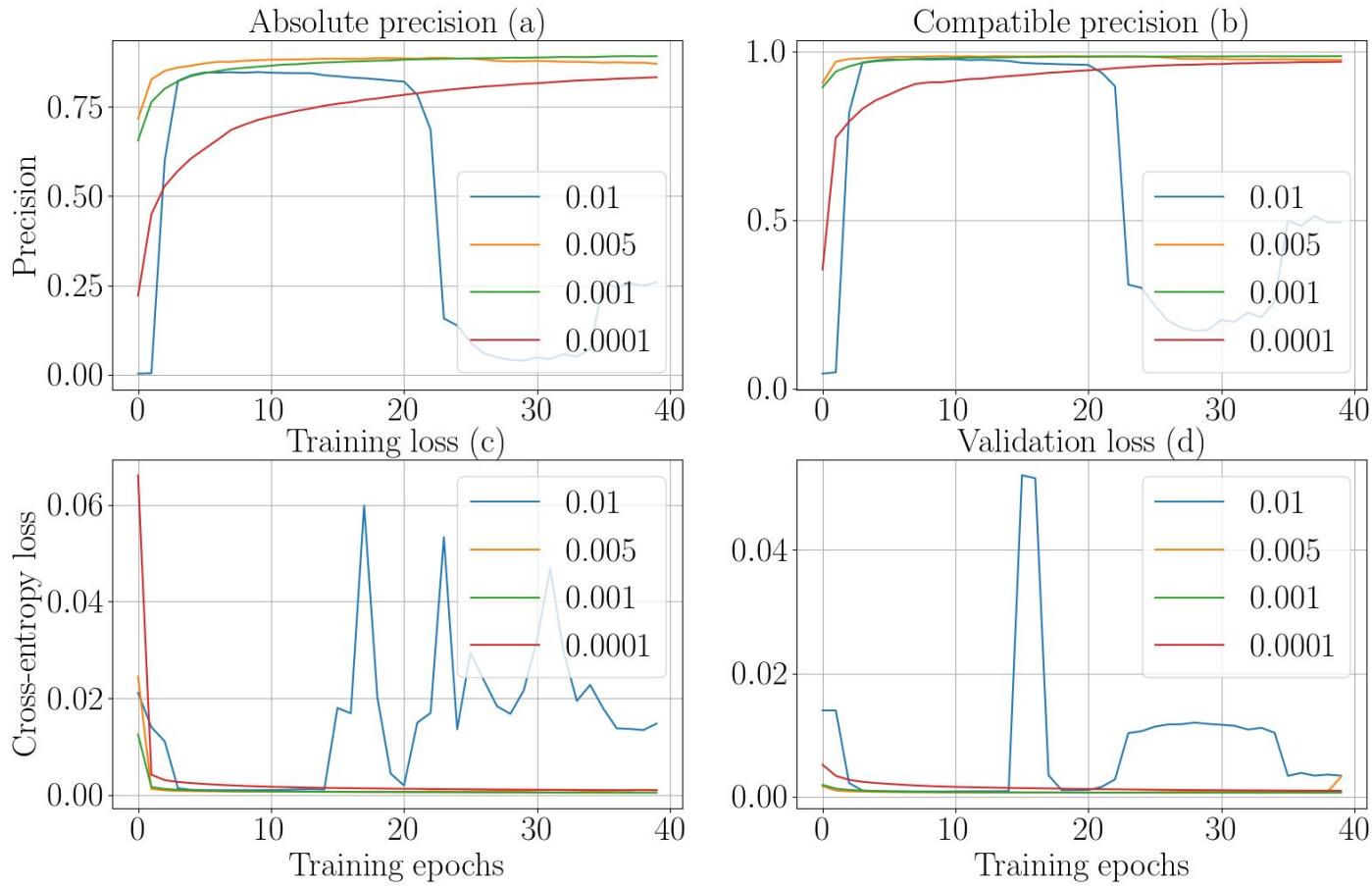
Distribution of number of tools in workflow paths



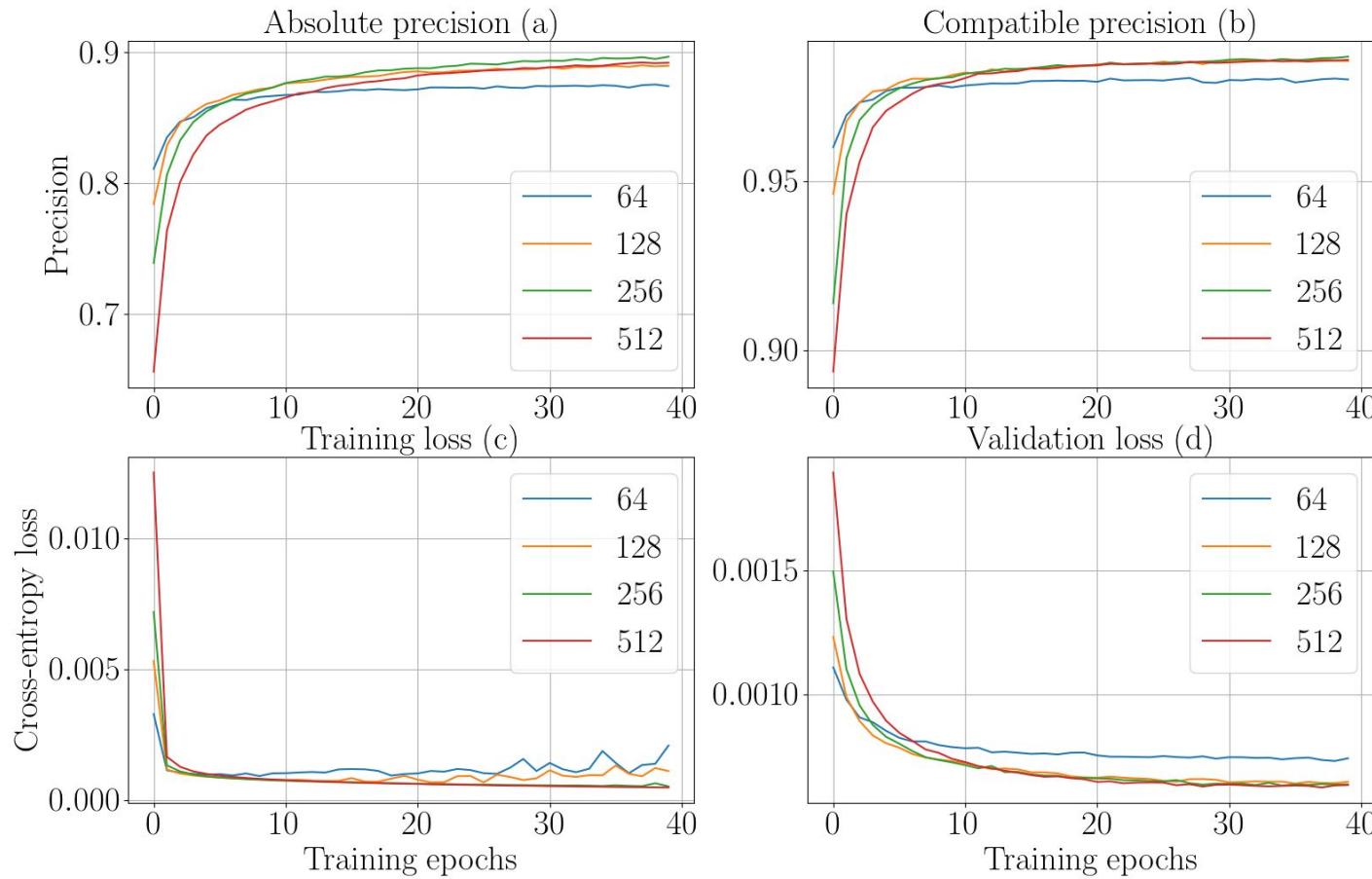
Precision and loss for various activations



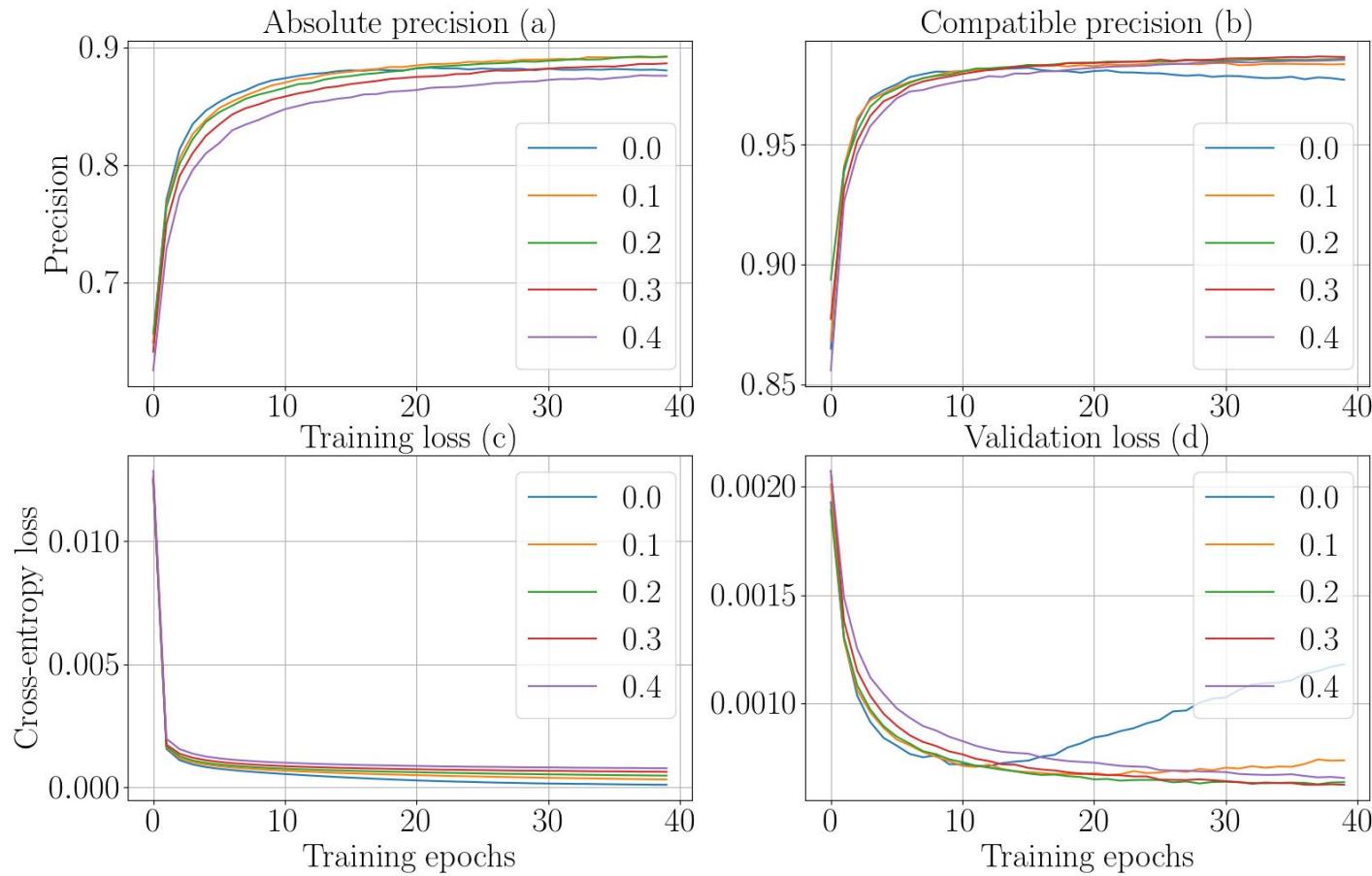
Precision and loss for various learning rates



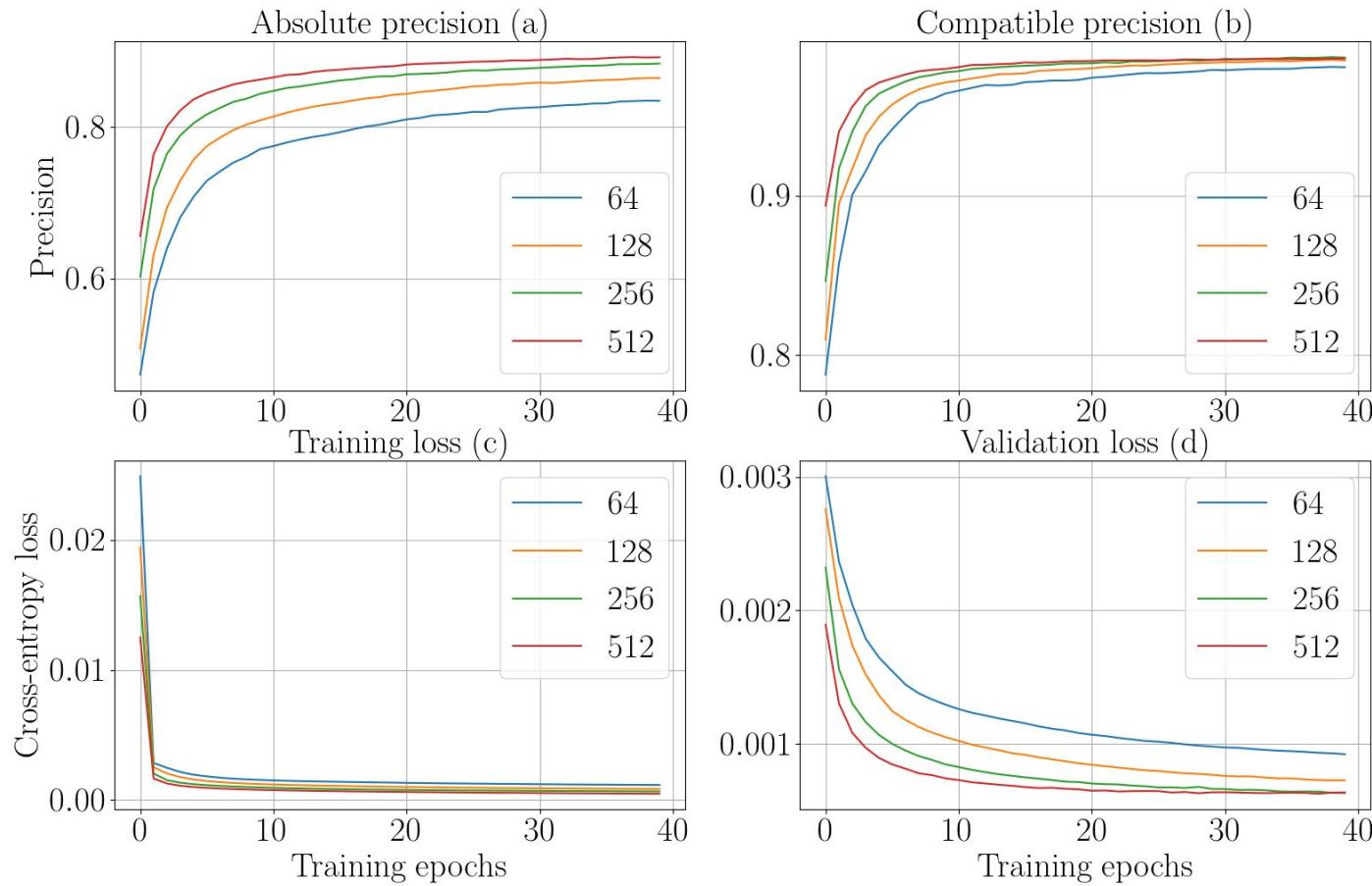
Precision and loss for various batch sizes



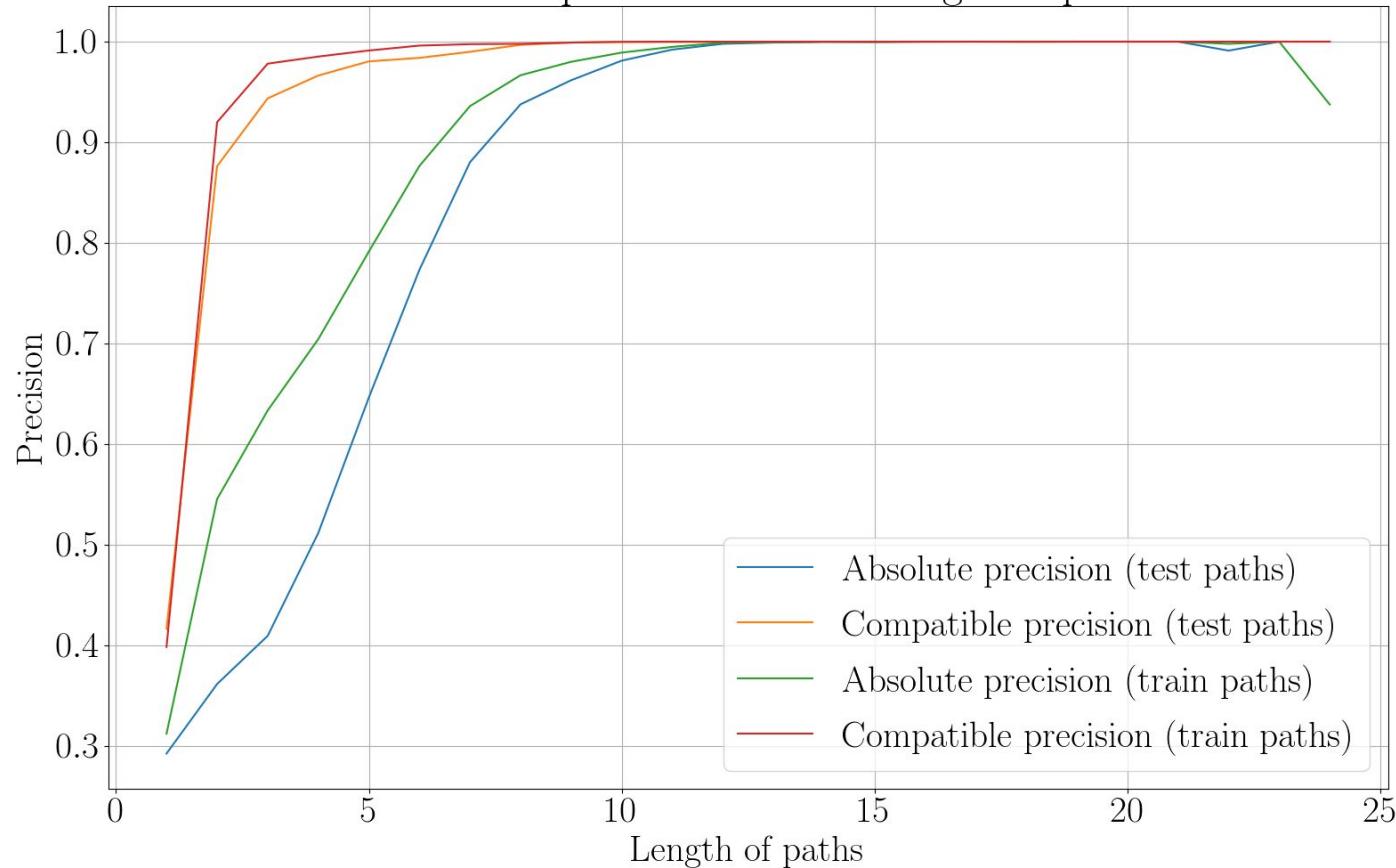
Precision and loss for various dropout values



Precision and loss for various number of memory units

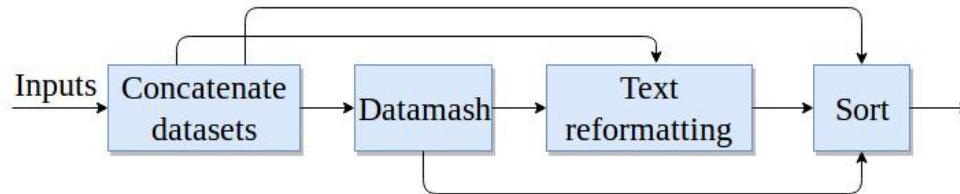


Variation of precision with the length of paths



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>

Bayesian network

- Explain workflows
- Predict missing nodes
- Compute joint and conditional probabilities
- High computational cost with large number of nodes
- Prediction by probabilistic network is a hard problem

<https://www.microsoft.com/en-us/research/uploads/prod/2004/01/Large-Sample-Learning-of-Bayesian-Networks-is-NP-Hard.pdf>
<https://www.sciencedirect.com/science/article/pii/000437029090060D>

Hidden markov models

- Transition and prior probabilities
- Transition matrix is large
- Current state depends only on previous state (first-order)
- Higher number of parameters
- Higher order markov models

<https://arxiv.org/pdf/q-bio/0505002.pdf>

<https://pdfs.semanticscholar.org/8463/dfee2b46fa813069029149e8e80cec95659f.pdf>

<http://mlg.eng.cam.ac.uk/zoubin/papers/ijprai.pdf>

Absolute and compatible top-1 and top-2 accuracies

