Large Dataset Source Code Classification

Checking capabilities of simple models and techniques

Vladimir Zolotov

December 15, 2020

Classification C++ source code files for many classes

- **<u>Old results</u>**:
- **57 AIZ**U problems/classes: 712 – 5099 solutions
  - Total 80,029 samples (80 : 20% split)
  - **94.72%** accuracy
    - Details are in the attached old charts in the appendix.

- **<u>New results:</u>**
- **536 AIZU problems** /classes with > 256 solutions
  - Total *395,870* samples (80 : 20% split), longest code 5113 tokens
  - **92.82% accuracy** at 29-th epoch
    - Each epoch runs 250 sec = 4 m 10s on single P100

- **1163 Atcoder problems**/classes with > 500 solutions,
  - Total *3,733,717* samples (80:20% split) longest code 4998 tokens
    - Solutions with code having >5000 tokens were excluded from classification
  - **89.91% accuracy** at 9-th epoch
    - Each epoch runs 3846 sec = 1h 4m 6s on single P100

- **<u>Conclusions:</u>**
  - Classification of both AIZU and Atcoder solutions can be done with high accuracy
    - Accuracy can be improved further by finer tuning DNNs and optimization

New vs Old classification techniques

Similarities

- Sequence of tokens model of source code (no comments, no identifiers)
  - Geert's tokenizer
- CNN with GlobalMax Pooling and Softmax classifier

Differences

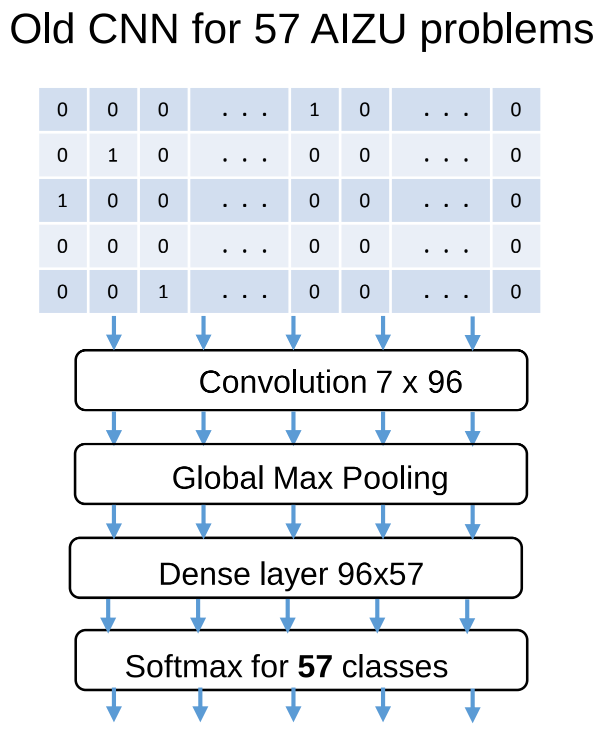
Old technique

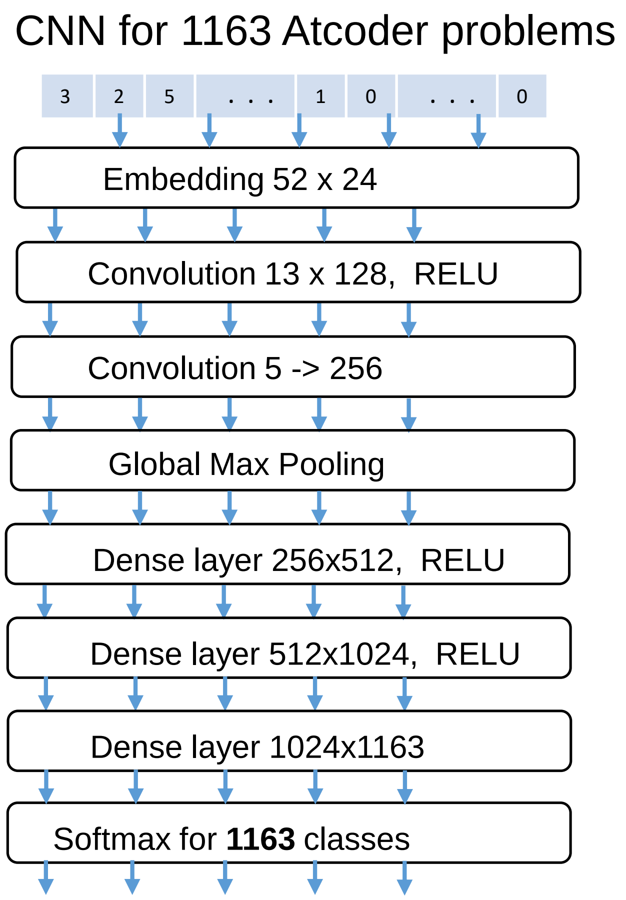
- 17 groups of combined tokens
  - H. Ohashi and Y. Watanobe, Convolutional Neural Network for Classification of Source Codes, 2019
- One-hot coding
- No embedding layer
- 4 layers:
  - 1x 1D convolution, Global Max Pool, 1x Dense, Softmax
- RMSPROP optimizer

New technique

- 52 tokens (no grouping)
  - Almost all operators and a few keywords
- Categorical coding
- Embedding layer
- Trainable embeddings
- 7 layers for Atcoder:
  - 2x 1D convolution, Global Max Pool, 3x Dense, Softmax
    - For AIZU only 6 layers: only 2 Dense
- ADAM optimizer

Old AIZU-57 and new Atcoder-1163 CNNs

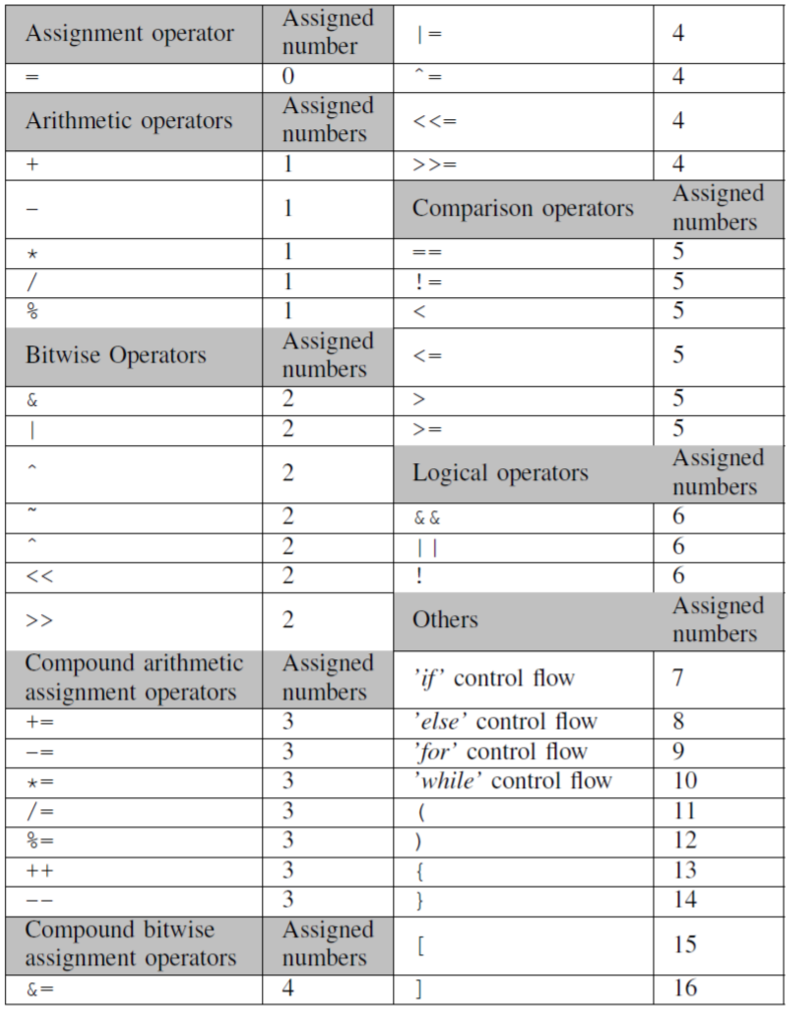




Old and New sequence elements

Old groups of tokens

ALL KEYWORDS AND TOKEN NUMBERS



Assignment operator	Assigned number	|=	4
=	0	^=	4
Arithmetic operators	Assigned numbers	<<=	4
+	1	>>=	4
−	1	Comparison operators	Assigned numbers
*	1	==	5
/	1	!=	5
%	1	<	5
Bitwise Operators	Assigned numbers	<=	5
&	2	>	5
|	2	>=	5
^	2	Logical operators	Assigned numbers
~	2	&&	6
^	2	||	6
<<	2	!	6
>>	2	Others	Assigned numbers
Compound arithmetic assignment operators	Assigned numbers	*'if'* control flow	7
+=	3	*'else'* control flow	8
−=	3	*'for'* control flow	9
*=	3	*'while'* control flow	10
/=	3	(	11
%=	3	)	12
++	3	{	13
−−	3	}	14
Compound bitwise assignment operators	Assigned numbers	[	15
&=	4	]	16

New tokens

- =, +, -, *, /, %
  - Assignment and arithmetic
- &, |, ^, ~, <<, >>
  - Bitwise Operators
- +=, -=, *=, /=, %=, ++, --
  - Compound arithmetic assignment
- &=, |=, ^=, <<=, >>=
  - Compound bitwise assignment
- ==, !=, <, <=, >, >=,
  - Comparison operators
- &&, ||, !
  - Logical operators
- (,), {, }, [,], ->
- if, else, for, while, switch,
- int, char, short, long, float, double, bool

Current work and Future Plans

- **<u>On-going work:</u>**
  - Similarity analysis for large datasets: AIZU-536 and Atcoder-1163
    - Applying similar modifications to Siamese version of CNNs used for AIZU-57 dataset
    - To be completed this year
    - Also expecting high accuracy
- **<u>Nearest plans:</u>**
  - Cross-language similarity analysis
    - Starting from C++ to Java using Geert's tokenizer
  - Using sequences of detailed tokens constructed from parse trees:
    - Distinguish between using * for multiplication and dereferencing, etc.
    - Variable and operator types (int vs float)
  - Develop and try batch technique for efficient processing parse trees by GPU

Appendix

Old charts on classification of AIZU 57 problems by sequence and bag of tokens techniques

Source Code Classification as Sequence of Tokens using Convolutional Neural Network

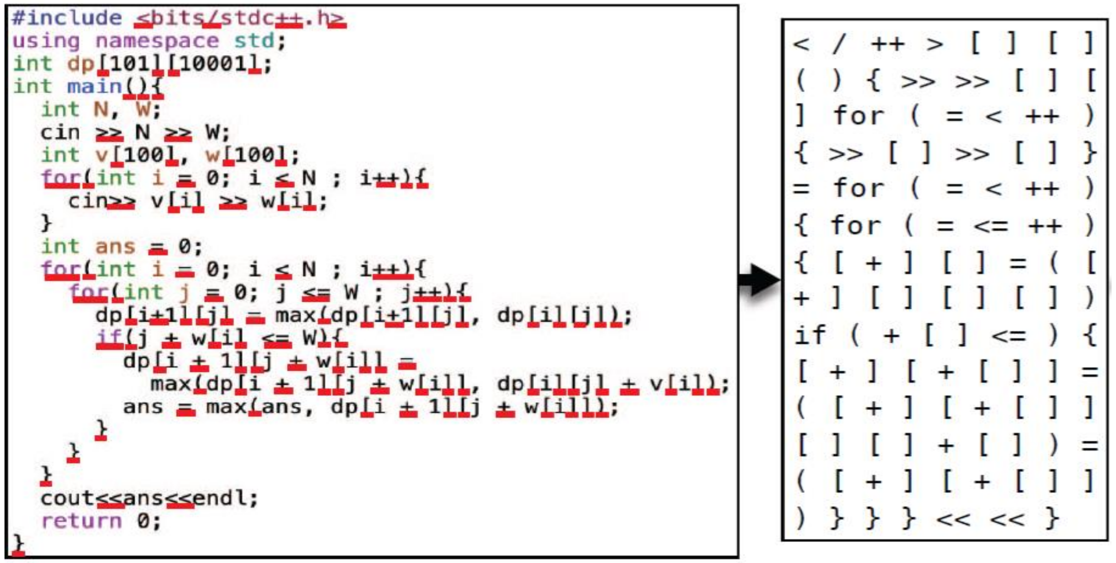
Checking capabilities of simple models and techniques

Vladimir Zolotov

October 13, 2020

Sequence of tokens model of source code

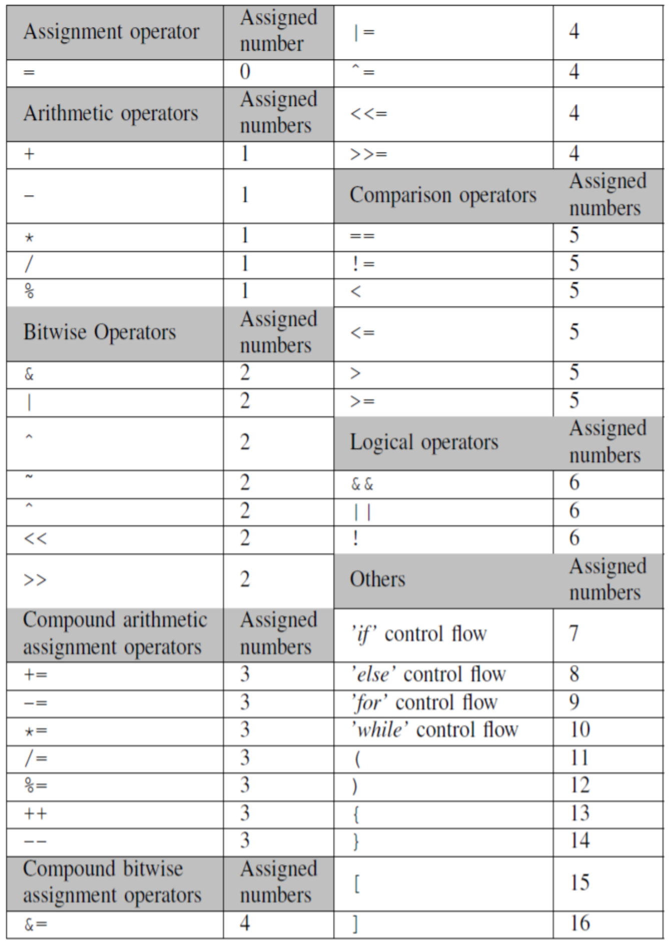
- Tokenization of source code
  - Geert Janseen tokenizer for C, C++
  - Very simple set of tokens:
    - Comments and macros are deleted
    - Groups of original language tokens are combined:
      +, -, *, /, % -> token #1
    - Hiroki Ohashi and Yutaka Watanobe, “Convolutional Neural Network for Classification of Source Codes”, 2019 Int. Symp. on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)
- One hot coding of each token
- Zero padding at the end to the length of the longest sequence



```
#include <bits/stdc++.h>
using namespace std;
int dp[101][10001];
int main(){
  int N, W;
  cin >> N >> W;
  int v[100], w[100];
  for(int i = 0; i < N ; i++){
    cin>> v[i] >> w[i];
  }
  int ans = 0;
  for(int i = 0; i < N ; i++){
    for(int j = 0; j <= W ; j++){
      dp[i+1][j] = max(dp[i+1][j], dp[i][j]);
      if(j + w[i] <= W){
        dp[i + 1][j + w[i]] =
          max(dp[i + 1][j + w[i]], dp[i][j] + v[i]);
        ans = max(ans, dp[i + 1][j + w[i]]);
      }
    }
  }
  cout<<ans<<endl;
  return 0;
}
```

```
< / ++ > [ ] [ ]
( ) { >> >> [ ] [
] for ( = < ++ )
{ >> [ ] >> [ ] }
= for ( = < ++ )
{ for ( = <= ++ )
{ [ + ] [ ] = ( [
+ ] [ ] [ ] [ ] )
if ( + [ ] <= ) {
[ + ] [ + [ ] ] =
( [ + ] [ + [ ] ]
[ ] [ ] + [ ] ) =
( [ + ] [ + [ ] ]
) } } } << << }
```

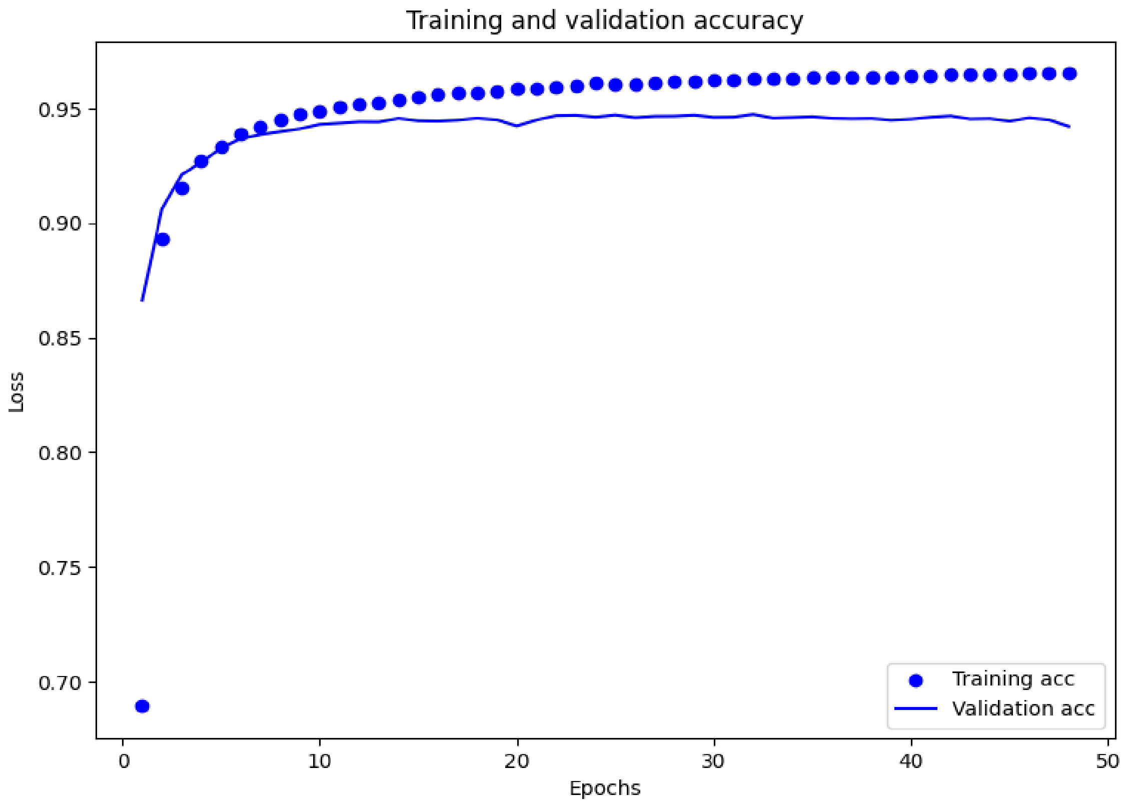
ALL KEYWORDS AND TOKEN NUMBERS

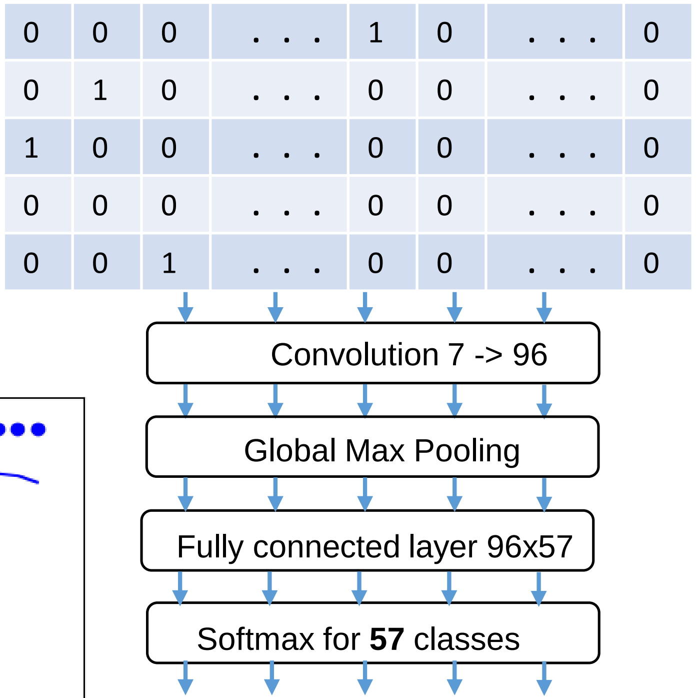


Assignment operator	Assigned number	|=	4
=	0	^=	4
Arithmetic operators	Assigned numbers	<<=	4
+	1	>>=	4
–	1	Comparison operators	Assigned numbers
*	1	==	5
/	1	!=	5
%	1	<	5
Bitwise Operators	Assigned numbers	<=	5
&	2	>	5
|	2	>=	5
^	2	Logical operators	Assigned numbers
~	2	&&	6
^	2	||	6
<<	2	!	6
>>	2	Others	Assigned numbers
Compound arithmetic assignment operators	Assigned numbers	*'if'* control flow	7
+=	3	*'else'* control flow	8
−=	3	*'for'* control flow	9
*=	3	*'while'* control flow	10
/=	3	(	11
%=	3	)	12
++	3	{	13
−−	3	}	14
Compound bitwise assignment operators	Assigned numbers	[	15
&=	4	]	16

Classification C++ source code files for many classes

- 57 classes are problems of AIZU dataset
  - Problems with 712 – 5099 *accepted* C++ solutions
  - 80% - 20% training/validation split
    - Training on 72024 samples, Validating on 18005 samples
- 3 Layer Neural network
  - 1D convolutional: Width = 7; 96 filters
  - Global max pooling
  - Fully connected layer with softmax: 96 x 57
- 94.72% accuracy at 25-th epoch





Source Code Classification by Bag of Tokens Technique

Checking capabilities of simple models and techniques

Vladimir Zolotov

Part of presentation on September 30, 2020

Bag of tokens model

- Tokenization of source code
    - Geert Janssen tokenizer for C, C++
  - Very simple set of 17 tokens:
    - Comments and macros are deleted
    - Groups of original language tokens are combined:

      +, -, *, /, %   -> token #1
      - "Convolutional Neural Network for Classification of Source Codes",

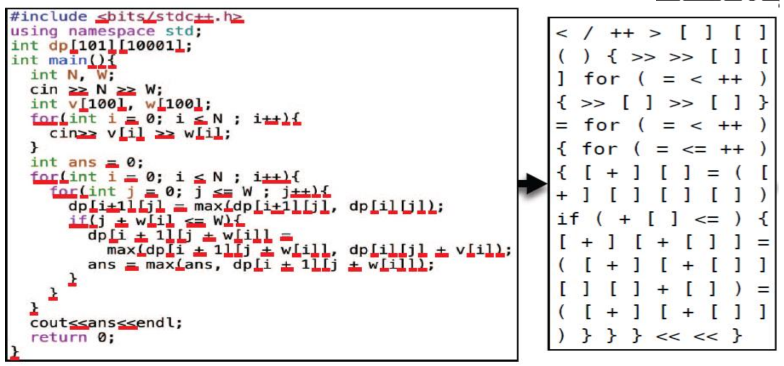
      Hiroki Ohashi and Yutaka Watanobe at 2019 Int. Symp. on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)
- Bag of tokens:
  - Make vector of number of tokens occurrences

=	+	-	*	. . .	{	}	[	]
5	10	0	0	. . .	4	4	25	25

  - Normalize it to get vector of token frequencies
    - *V = W/sqrt(W*W)*

=	+	-	*	. . .	{	}	[	]
0.13	0.21	0	0	. . .	0.12	0.12	0.47	0.47

- Advantages:
  - Invariant to many types of code transformations:
    - Statement permutations, code factorization, etc.
  - Many other models are not invariant to many code transformations preserving its algorithms



```
#include <bits/stdc++.h>
using namespace std;
int dp[101][10001];
int main(){
  int N, W;
  cin >> N >> W;
  int v[100], w[100];
  for(int i = 0; i < N ; i++){
    cin>> v[i] >> w[i];
  }
  int ans = 0;
  for(int i = 0; i < N ; i++){
    for(int j = 0; j <= W ; j++){
      dp[i+1][j] = max(dp[i+1][j], dp[i][j]);
      if(j + w[i] <= W){
        dp[i + 1][j + w[i]] =
          max(dp[i + 1][j + w[i]], dp[i][j] + v[i]);
        ans = max(ans, dp[i + 1][j + w[i]]);
      }
    }
  }
  cout<<ans<<endl;
  return 0;
}
```

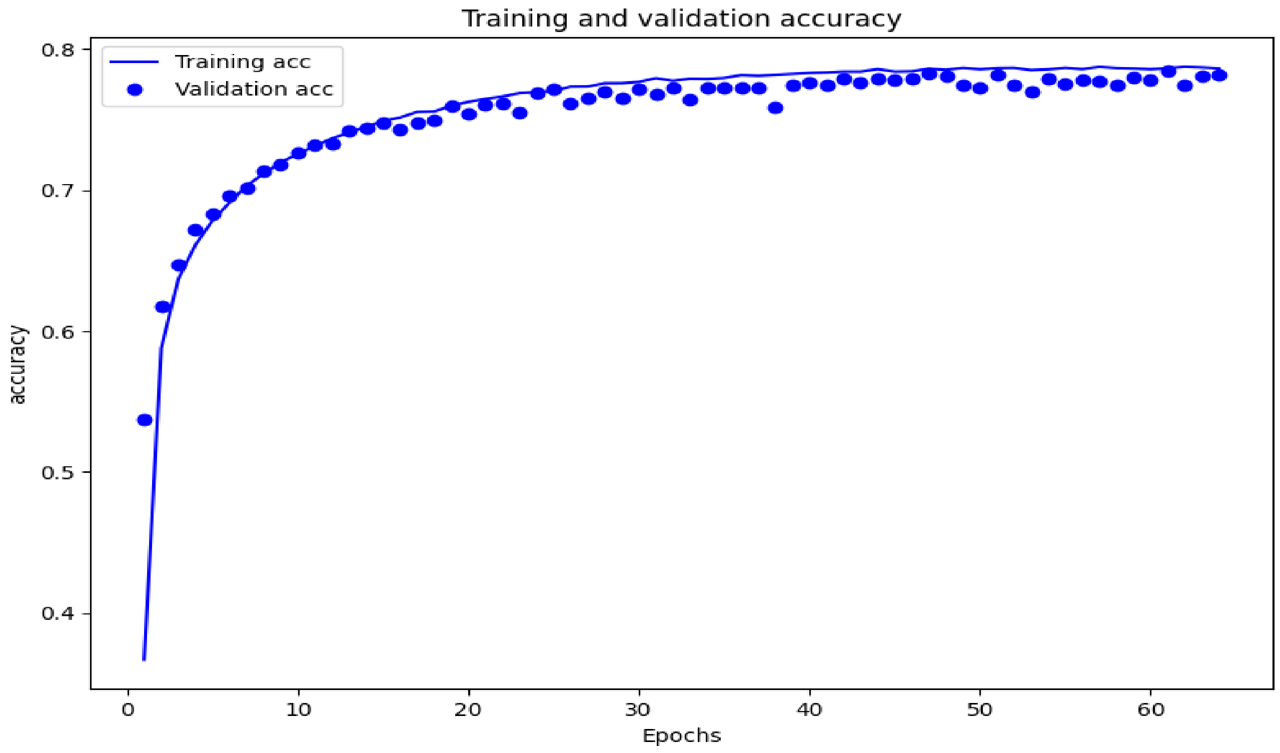
```
< / ++ > [ ] [ ]
( ) { >> >> [ ] [
] for ( = < ++ )
{ >> [ ] >> [ ] }
= for ( = < ++ )
{ for ( = <= ++ )
{ [ + ] [ ] = ( [
+ ] [ ] [ ] [ ] )
if ( + [ ] <= ) {
[ + ] [ + [ ] ] =
( [ + ] [ + [ ] ]
[ ] [ ] + [ ] ) =
( [ + ] [ + [ ] ]
) } } } << << }
```

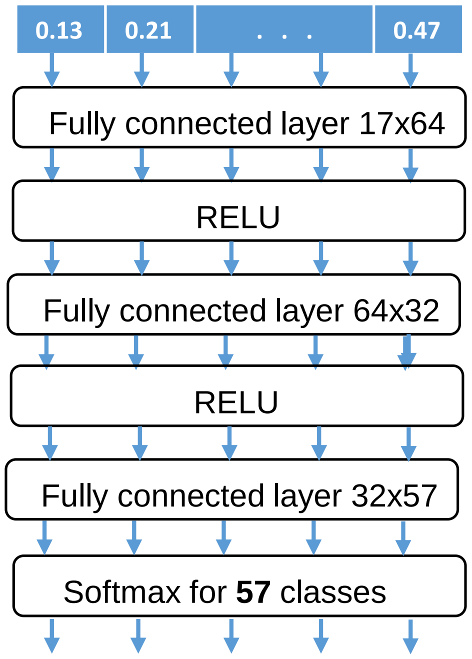
ALL KEYWORDS AND TOKEN NUMBERS

Assignment operator	Assigned number	|=	4
=	0	^=	4
Arithmetic operators	Assigned numbers	<<=	4
+	1	>>=	4
−	1	Comparison operators	Assigned numbers
*	1	==	5
/	1	!=	5
%	1	<	5
Bitwise Operators	Assigned numbers	<=	5
&	2	>	5
|	2	>=	5
^	2	Logical operators	Assigned numbers
~	2	&&	6
^	2	||	6
<<	2	!	6
>>	2	Others	Assigned numbers
Compound arithmetic assignment operators	Assigned numbers	*'if'* control flow	7
+=	3	*'else'* control flow	8
−=	3	*'for'* control flow	9
*=	3	*'while'* control flow	10
/=	3	(	11
%=	3	)	12
++	3	{	13
−−	3	}	14
Compound bitwise assignment operators	Assigned numbers	[	15
&=	4	]	16

Classification C++ source code files for many classes

- 57 classes are problems of AIZU dataset
  - Problems with 712 – 5099 *accepted* C++ solutions
    - Most problems have <1500 solutions
  - 80% - 20% training/validation split
    - Training on 72024 samples, Validating on 18005 samples
- 3 Layer Neural network
  - Fully connected layers: 17x64, 64x32, 32x57,
    - RELU and softmax activations
- 79.47% accuracy at epoch 61
  - Probability of random guess is only ~ 2%





0.13
0.21
. . .
0.47
Fully connected layer 17x64
RELU
Fully connected layer 64x32
RELU
Fully connected layer 32x57
Softmax for **57** classes