Subtask 1

Overview

Command Line Arguments

We have used the approach where the user can enter a interpreter (basic) and provide commands in a continous way.

All commands defined below follow , except for one constraint that *arrow keys must not be pressed*

To enter the interpreter type `./ipl` and just type `exit` to exit. Also, `clear` can be used to clear the screen.

Also, `...` can be use to repeat/augment last command.

Convolution

```
#include "convolution.h"
```

Command Format
Output on console > `function matrix_file kernel_file`
Saving on file > `function matrix_file kernel_file output_file`

- Direct convolution

Task	Function
Convolution without padding	`conv`
Convolution with padding	`conv_pad`

Task	Function
Cross-correlation without padding	`cross`
Cross-correlation with padding	`cross_pad`

- Matrix multiplication

Task	Function
Convolution without padding	`conv_mult`
Convolution with padding	`conv_mult_pad`
Cross-correlation without padding	`cross_mult`
Cross-correlation with padding	`cross_mult_pad`

Library (In code)

- Direct Convolution/Cross-correlation

```
vector<vector<float>> directConvolution(vector<vector<float>> kernel,
vector<vector<float>> matrix, bool convolution, bool padding=false)
```

- Convolution/Cross-correlation by matrix-multiplication

```
vector<vector<float>> convolutionByMultiplication(vector<vector<float>>
kernel, vector<vector<float>> matrix, bool convolution, bool padding=false)
```

Activation

```
    #include "activation.h"
```

- ***relu (Rectified Linear Units)***
  - *Command Format* (On console)
    Output on console > `relu matrix_file num_rows`
    Saving on file > `relu matrix_file num_rows output_file`
  - *Library* (In code)

    ```
    vector<vector<float>> relu(<vector<vector<{float, int}>> matrix);
    ```

- ***tanh (Hyperbolic)***

- *Command Format* (On console)
  Output on console > `tanh matrix_file num_rows`
  Saving on file > `tanh matrix_file num_rows output_file`

- *Library* (In code)

```
vector<vector<float>> tanh(<vector<vector<{float, int}>> matrix);
```

- ***Sigmoid***

  - *Command Format* (On console)
    Output on console > `sigmoid vector_file`
    Saving on file > `sigmoid vector_file output_file`

  - *Library* (In code)

```
vector<float> sigmoid(vector<float> arr);
```

- ***Softmax***

  - *Command Format* (On console)
    Output on console > `softmax vector_file`
    Saving on file > `softmax vector_file output_file`

  - *Library* (In code)

```
vector<float> softmax(vector<float> arr);
```

Pooling

```
#include "pool.h"
```

- ***Max Pool***
  - *Command Format* (On console)
    Output on console > `max_pool matrix_file filter_size stride`
    Saving on file > `max_pool matrix_file filter_size stride output_file`

  - *Library* (In code)

```
vector<vector<{int, float}>> maxPool(vector<vector<{int, float}>> matrix,
int filterSize = 2, int stride = 2 );
```

- ***Average Pool***
  - *Command Format* (On console)
    Output on console > `avg_pool matrix_file filter_size stride`
    Saving on file > `avg_pool matrix_file filter_size stride output_file`

  - *Library* (In code)

```
vector<vector<{int, float}>> avgPool(vector<vector<{int, float}>> matrix,
int filterSize = 2, int stride = 2 );
```

Other Helpful Functions

Matrix View

To view the result in a formatted form on the console. Just for a review.

Command Format
Matrix > `view filename num_rows`
Square Matrix > `view_square filename`

Save Previous

To augment the previous command (output on console) and saving the output to a file

Command Format
To repeat last command > `...`
To augment last command (save only) > `... output_file`