Overview

Pikafish is a free, powerful UCI xiangqi engine
derived from Stockfish. Pikafish is not a complete xiangqi program and requires a
UCI-compatible graphical user interface (GUI) (e.g. VinXiangQi or BHGUI) in order
to be used comfortably. Read the documentation for your GUI of choice for information
about how to use Pikafish with it.

The Pikafish engine features the efficiently updatable neural network (NNUE) based
evaluation for xiangqi, which is by far the strongest. The strongest network can be
downloaded from our official website. The NNUE evaluation benefits from the vector
intrinsics available on most CPUs (sse2, avx2, neon, or similar).

Files

This distribution of Pikafish consists of the following files:

- README.md,
  the file you are currently reading.
- Copying.txt,
  a text file containing the GNU General Public License version 3.
- AUTHORS,
  a text file with the list of authors for the official Stockfish project.
- src,
  a subdirectory containing the full source code, including a Makefile
  that can be used to compile Pikafish on Unix-like systems.

The UCI protocol and available options

The Universal Chess Interface (UCI) is a standard protocol used to communicate with
a chess engine, and is the recommended way to do so for typical graphical user interfaces
(GUI) or chess tools. Pikafish implements the majority of its options as described
in the UCI protocol.

Developers can see the default values for UCI options available in Pikafish by typing
`./pikafish uci` in a terminal, but the majority of users will typically see them and
change them via a chess GUI. This is a list of available UCI options in Pikafish:

- **Threads**

  The number of CPU threads used for searching a position. For best performance, set
  this equal to the number of CPU cores available.

- **Hash**

  The size of the hash table in MB. It is recommended to set Hash after setting Threads.

- **Clear Hash**

  Clear the hash table.

- **Ponder**

  Let Pikafish ponder its next move while the opponent is thinking.

- **MultiPV**

  Output the N best lines (principal variations, PVs) when searching.
  Leave at 1 for best performance.

- **EvalFile**

  The name of the file of the NNUE evaluation parameters. Depending on the GUI the filename might have to include the full path to the folder/directory that contains the file. Other locations, such as the directory that contains the binary and the working directory, are also searched.

- **UCI_AnalyseMode**

  An option handled by your GUI.

- **UCI_ShowWDL**

  If enabled, show approximate WDL statistics as part of the engine output.
  These WDL numbers model expected game outcomes for a given evaluation and game ply for engine self-play at fishtest LTC conditions (60+0.6s per game).

- **Move Overhead**

  Assume a time delay of x ms due to network and GUI overheads. This is useful to avoid losses on time in those cases.

- **Slow Mover**

  Lower values will make Pikafish take less time in games, higher values will make it think longer.

- **nodestime**

  Tells the engine to use nodes searched instead of wall time to account for elapsed time. Useful for engine testing.

- **Debug Log File**

  Write all communication to and from the engine into a text file.

For developers the following non-standard commands might be of interest, mainly useful for debugging:

- **bench *ttSize threads limit fenFile limitType***

  Performs a standard benchmark using various options. The signature of a version (standard node count) is obtained using all defaults. `bench` is currently `bench 16 1 13 default depth`.

- **compiler**

  Give information about the compiler and environment used for building a binary.

- **d**

  Display the current position, with ascii art and fen.

- **eval**

  Return the evaluation of the current position.

- **export_net [filename]**

  Exports the currently loaded network to a file.
  If the currently loaded network is the embedded network and the filename
  is not specified then the network is saved to the file matching the name
  of the embedded network, as defined in evaluate.h.
  The filename parameter is required and the network is saved into that file.

- **flip**

  Flips the side to move.

A note on NNUE evaluation

This approach assigns a value to a position that is used in alpha-beta (PVS) search
to find the best move. The NNUE evaluation computes this value with a neural network
based on basic inputs (e.g. piece positions only). The network is optimized and trained
on the evaluations of millions of positions at moderate search depth.

The NNUE evaluation was first introduced in shogi, and ported to Stockfish afterward.
It can be evaluated efficiently on CPUs, and exploits the fact that only parts
of the neural network need to be updated after a typical chess move.
The nodchip repository provided the first
version of the needed tools to train and develop the NNUE networks. Today, more
advanced training tools are available in
the nnue-pytorch repository,
while data generation tools are available in
a dedicated branch.

We change the feature sets and port the HalfKAv2_hm structure to xiangqi
and rename it as HalfKAv2_xq.

On CPUs supporting modern vector instructions (avx2 and similar), the NNUE evaluation
results in much stronger playing strength, even if the nodes per second computed by
the engine is somewhat lower (roughly 80% of nps is typical).

Notes:

1. the NNUE evaluation depends on the Pikafish binary and the network parameter file
   (see the EvalFile UCI option). Not every parameter file is compatible with a given
   Pikafish binary, but the default value of the EvalFile UCI option is the name of a
   network that is guaranteed to be compatible with that binary.
2. to use the NNUE evaluation, the additional data file with neural network parameters
   needs to be available. The filename for the default (recommended) net can be found as

the default value of the `EvalFile` UCI option, with the format `pikafish.nnue`. This file can be downloaded from `http://test.pikafish.org`.

Large Pages

Pikafish supports large pages on Linux and Windows. Large pages make the hash access more efficient, improving the engine speed, especially on large hash sizes. Typical increases are 5..10% in terms of nodes per second, but speed increases up to 30% have been measured. The support is automatic. Pikafish attempts to use large pages when available and will fall back to regular memory allocation when this is not the case.

Support on Linux

Large page support on Linux is obtained by the Linux kernel transparent huge pages functionality. Typically, transparent huge pages are already enabled, and no configuration is needed.

Support on Windows

The use of large pages requires "Lock Pages in Memory" privilege. See Enable the Lock Pages in Memory Option (Windows) on how to enable this privilege, then run RAMMap to double-check that large pages are used. We suggest that you reboot your computer after you have enabled large pages, because long Windows sessions suffer from memory fragmentation, which may prevent Pikafish from getting large pages: a fresh session is better in this regard.

Compiling Pikafish yourself from the sources

Pikafish has support for 32 or 64-bit CPUs, certain hardware instructions, big-endian machines such as Power PC, and other platforms.

On Unix-like systems, it should be easy to compile Pikafish directly from the source code with the included Makefile in the folder `src`. In general it is recommended to run `make help` to see a list of make targets with corresponding descriptions.

```
cd src
make help
make build ARCH=x86-64-modern
```

When not using the Makefile to compile (for instance, with Microsoft MSVC) you need to manually set/unset some switches in the compiler command line; see file *types.h* for a quick reference.

When reporting an issue or a bug, please tell us which Pikafish version and which compiler you used to create your executable. This information can be found by typing the following command in a console:

```
./pikafish compiler
```

Understanding the code base and participating in the project

Pikafish's improvement over the last decade has been a great community effort. There are a few ways to help contribute to its growth.

Donating hardware

Improving Pikafish requires a massive amount of testing. You can donate your hardware resources by installing the
Fishtest Worker
and view the current tests on Fishtest.

Improving the code

If you want to help improve the code, there are several valuable resources:

- In this wiki, many techniques used in
  Pikafish are explained with a lot of background information.
- The section on Stockfish
  describes many features and techniques used by Pikafish. However, it is generic rather than being focused on Stockfish's precise implementation. Nevertheless, a helpful resource.
- The latest source can always be found on GitHub.
  Discussions about Pikafish take place these days mainly in the
  Pikafish 分布式训练 / Vin象棋连线 QQ group.
  The engine testing is done on Fishtest.
  If you want to help improve Pikafish, please read this guideline
  first, where the basics of Pikafish development are explained.

Terms of use

Pikafish is free, and distributed under the **GNU General Public License version 3** (GPL v3). Essentially, this means you are free to do almost exactly what you want with the program, including distributing it among your friends, making it available for download from your website, selling it (either by itself or as part of some bigger software package), or using it as the starting point for a software project of your own.

The only real limitation is that whenever you distribute Pikafish in some way, you MUST always include the license and the full source code (or a pointer to where the source code can be found) to generate the exact binary you are distributing. If you make any changes to the source code, these changes must also be made available under the GPL v3.

For full details, read the copy of the GPL v3 found in the file named *Copying.txt*.

Goal of the project

The current goal of the project is to make Pikafish suppress all commercial engines and become the top 1. Terminating the domination of those commercial engines like AlphaCat, BugChess, and Cyclone in xiangqi. Making the strongest xiangqi engine free and open source to everyone.

I believe if Stockfish can do that for chess, Pikafish can do that for xiangqi.

Hall of shame

The following individuals and organizations are prohibited from using the nnue weights file derived from this program due to their unfriendly actions, activities, and license violations（鉴于近期出现的种种不友好的行为，以下个人和组织将被禁止使用由本程序引申出的所有 nnue 权重文件）：

- AlanThinker (鹏飞象棋)
- 国圣
- 飞风追云