## Typesetting Neural Nets using XY-pic

Ross Moore*
Mathematics Department
Macquarie University
NSW 2109, Australia
ross@mpce.mq.edu.au

### Abstract

This note gives an example of how to use XY-pic's "Graph Combinator" feature to specify the type of diagram that is frequently used to display Neural Networks.

Look at the following diagram[1] and attempt to describe it in words, capturing the essential structure.

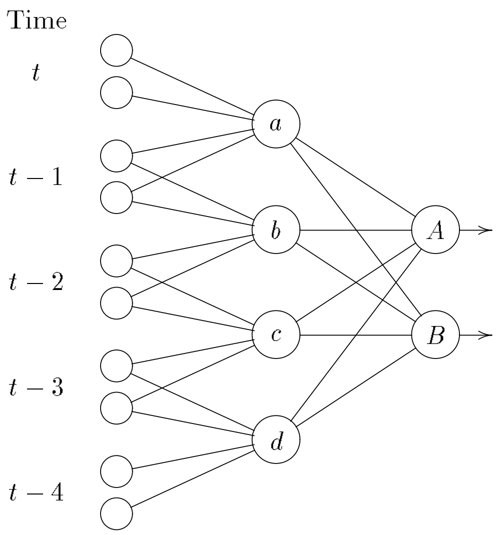


**Figure 1**: Example of a Neural Network

Here is one possible attempt.

> *There are 3 types of object. Two objects, called A and B, are positioned one above the other; from each an arrow points to the right, away from the main part of the diagram.*
>
> *To the left of these are four more objects, each connected by lines to both A and B. They are positioned vertically above one another and labelled a, b, c, d respectively.*
>
> *Further to the left are five more objects, again placed above each other, but so that the inner ones are located intermediate between the objects of the second type. These are more complicated, being a pair of sub-objects each connected to the nearest two of the second type, except for the outer ones which connect only to the single nearest object of the second type. Furthermore, each pair has a label indicating a time-value, from $t$ decreasing in unit steps to $(t-4)$.*
>
> *Objects and sub-objects are displayed as circles surrounding the single-letter names where applicable.*

Now look at the following segment of code, which was actually used to specify figure 1.

```
$$\xy\xygraph{!{<5pc,0pc>:<0pc,3.3pc>::}
 [] !{\OutNeuron A}
 [d]!{\OutNeuron B}
"A"[lu]!{\NeuronAB a}
    [d]!{\NeuronAB b}
    [d]!{\NeuronAB c}
    [d]!{\NeuronAB d}
"a"[u(.5)l]
    (?!{\Time0},?!{\biNeuron1 a})
[d] (?!{\Time1},?!{\biNeuron2{ab}})
[d] (?!{\Time2},?!{\biNeuron2{bc}})
[d] (?!{\Time3},?!{\biNeuron2{cd}})
[d] (?!{\Time4},?!{\biNeuron1 d})
}\endxy $$
```

Some syntax needs to be explained, but basically the three types of object are represented by `\OutNeuron`, `\NeuronAB` and `\biNeuron`, each occurring within `!{...}` and accompanied by a simple specification of its name or its links to other objects. Given that `[l]`, `[d]` and `[lu]` essentially mean "move left", "move down" and "move left and up" respectively and `"A"` means "move to where $A$ is", then the code is seen to correspond closely to the verbal description. Including a factor allows moves by fractional amounts, e.g. `[u(.5)l]` moves a half-step up and full step to the left.

Initially `!{<5pc,0pc>:<0pc,3.3pc>::}` establishes basis directions and lengths for the "hops" using `[l]`, `[d]`, etc. In particular `<5pc,0pc>:` establishes that `[r]` will subsequently move 5pc to the right with no vertical shift, while `<0pc,3.3pc>::` establishes `[u]` will move 3.3pc upward with no horizontal shift. Then `[l]`, `[d]` move opposite to `[r]` and `[u]` respectively. Composite hops, such as `[ddr]`, are interpreted by adding the displacements for the occurrences of `u,d,r,l`.

The above meanings are established by the `\xygraph` command which interprets its single argument as a collection of moves and actions to take at

* work was partially supported by the Australian Research Council and through a research agreement with the Digital Equipment Corporation.

[1] thanks also to Peter Vamplew and La Monte Yarroll, University of Tasmania, for this example.

the resulting locations. For a complete description of the possibilities, see "Graph Combinator" feature in the XY-pic Reference Manual[2]. Constructions of the form `!{...}` are used to execute portions of XY-pic kernel code and higher level user-defined code. Parentheses `(...)` enclose a comma-separated list of actions in which `?` refers to the position immediately prior to encountering the list.

Now TEX cannot fully interpret the above code until we have defined macros `\OutNeuron`, `\NeuronAB`, `\biNeuron` and `\Time`. Those used in figure 1 will now be described. They involve simple TEX programming constructions, to generate fragments of XY-pic code which precisely specify the lines, circles, arrowheads and pieces of text which must come together to produce the picture as typeset above.

- Simplest is `\OutNeuron`

  ```
  \def\OutNeuron#1{%
   \Neuron{#1}="#1"\Output}
  \def\Output{\ar+(.35,0)}
  ```

  which uses `#1` to name the current position. The `\ar` draws an arrow of length .35 units.

- The neuron itself is drawn by

  ```
  \def\Neuron#1{\POS
   *{#1}*++[o]{}*+\frm{o}}
  ```

  which typesets the contents of `#1`, then ignores its size by following it with a circular frame around an empty object of fixed circular size (with radius = $2 \times$ *objectmargin*). The initial call to `\POS`, perhaps the most important of all the primitive commands within XY-pic, initiates parsing of the subsequent tokens.

- The macro `\NeuronAB` similarly typesets using `\Neuron`, but also includes links to $A$ and $B$. These are simply an `\ar` without arrow-tips:

  ```
  \def\NeuronAB#1{\Neuron{#1}="#1"
   \Link"A",\Link"B"}
  \def\Link{\ar @{-}}
  ```

- More complicated is `\biNeuron` since there are (at least) two types.

  ```
  \def\biNeuron#1#2{%
   \ifcase#1\def\next{\relax}\or
    \def\next{\leafs#2}\or
    \def\next{\bileafs#2}\else
    \def\next{\relax}\fi\next}
  \def\leafs#1{%
   \save\uNeuron\Link"#1"\restore
   \save\dNeuron\Link"#1"\restore}
  \def\bileafs#1#2{\save\uNeuron
    \Link"#1",\Link"#2"\restore \save
    \dNeuron\Link"#1",\Link"#2"\restore}
  ```

  Typesetting done by `\uNeuron` and `\dNeuron` is similar to `\Neuron` but incorporate a displacement before dropping the circular frame:

  ```
  \def\uNeuron{\POS
   +(0,.2)*++[o]{}*\frm{o}}
  \def\dNeuron{\POS
   -(0,.2)*++[o]{}*\frm{o}}
  ```

- Finally the time labels are placed using ...

  ```
  \def\Timelabel{Time}
  \def\Time#1{\ifcase#1
   \POS-(.5,0)*{t}\save
   \POS+(0,.5)*\txt{\Timelabel}\restore
   \else\POS-(.5,0)*{t-#1}\relax\fi}
  ```

  This both constructs the label to be typeset and positions it relative to the location of the `\biNeuron`. Notice the special treatment of the case at time $t$.

Only the following macros actually place any marks on the page, the others handling the logic determining what should be placed and where to place it: `\uNeuron`, `\dNeuron`, `\Neuron`, `\Output`, `\Link`, `\Time`, `\Timelabel`. Thus details in the picture may be changed by redefining these to place different things, while the logic governing the overall layout remains the same.

On the other hand it is easy to change parts of the code for figure 1, e.g. add more neurons or remove some or link them differently, to obtain new diagrams while maintaining a consistent style. For example it is not always necessary to have labels and frequently networks are drawn with horizontal layers.

The diagram in figure 2 was specified using the same code as previously, except for setting-up the basis with `!{<0pc,4pc>:<-3.3pc,0pc>::}`, `\def\Output{\ar+(.4,0)}\def\Timelabel{}` and `\def\Neuron#1{\POS*+[o],*+\frm{o}}` so as to be smaller and omit the labels.

Labels can be easily added to the links. In the following code the characters `^` `_` and `|` are used to place the label "above", "below" or at a "break"

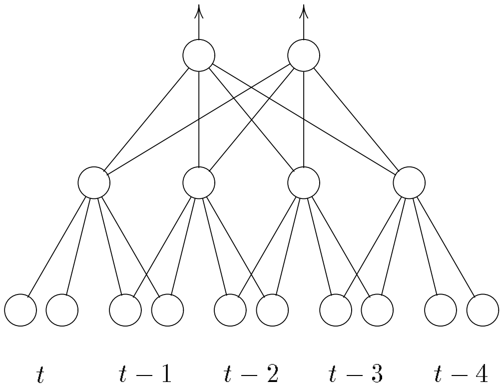


**Figure 2**:

in the middle of the link. A specification such as `{^(.3){`*label*`}}` would position the label the given fraction along the link, rather than halfway.

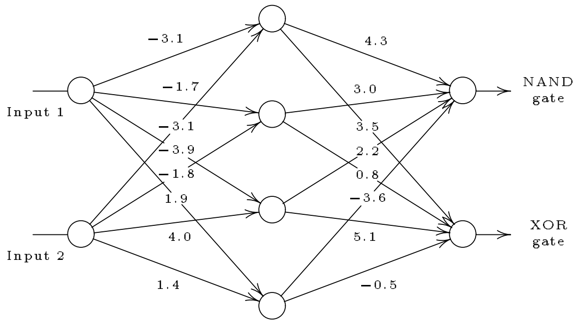


**Figure 3**: Neural network for Boolean operations

```
\def\Neuron{\POS*+[o]{},*=<10pt>\frm{o}}
\def\Output#1{\ar +(.25,0)*!L
 \txt<1cm>{\tiny #1}}
\def\Input#1{\POS{\ar @{-}-(.25,0)}
 ,="1"*++!UR\txt{\tiny #1}}

\def\OutNeuron#1#2{\Neuron="#1"
 \Output{#2}}
\def\InNeuron#1#2#3#4#5{%
 \expandafter\POS\Neuron;p\Input{#1}\POS p
 \save\ar"a"#2\restore\save\ar"b"#3
 \restore\save\ar"c"#4\restore
 \save\ar"d"#5\restore }
\def\Hidden#1#2#3{\Neuron="#1"\save
  \ar"A"#2\restore \save\ar"B"#3
 \restore}
%
\def\labelbox#1{\object+<3pt>
 {\scriptscriptstyle#1}}
```

```
$$\xy\xygraph{!{<6pc,0pc>:<0pc,3pc>::@i}
   []!{+(0,.25)}!{\OutNeuron A{NAND gate}}
  [d]!{-(0,.50)}!{\OutNeuron B{XOR gate}}
"A"!{+(-1,.75)}
       !{\Hidden a{^{4.3}}{|{3.5}}}
     [d]!{\Hidden b{^{3.0}}{|{0.8}}}
     [d]!{\Hidden c{|{2.2}}{_{5.1}}}
     [d]!{\Hidden d{|{-3.6}}{_{-0.5}}}
"b"[l]!{+(0,.25)}!{\InNeuron{Input 1}%
 {^{-3.1}}{^{-1.7}}{|{-3.9}}{|{1.9}}}
   [d]!{-(0,.50)}!{\InNeuron{Input 2}%
 {|{-3.1}}{|{-1.8}}{_{4.0}}{_{1.4}}}
}\endxy $$
```

The `\labelbox` command here determines the formatting for labels on the links, including the spacing around those that occur as "breaks".

Many neural networks can be typeset without the need to explicitly assign names to the nodes. For example the layered network in figure 4 was typeset using the following code.

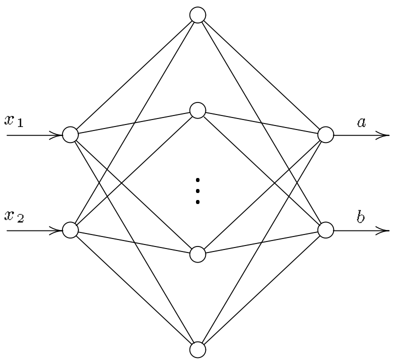


**Figure 4**: Layered neural net

```
\def\OutNeuron#1{\Neuron @+
  \Output{#1}}
\def\InNeuron#1{\Neuron @+
  \Input{#1}}
\def\Hidden{\Neuron
 ;@@{\connect\dir{-}}\POS p}

\def\Neuron{\POS*+[o]{},*\frm{o}}
\def\Output#1{\ar +(.5,0)^-{#1}}
\def\Input#1{\ar @{<-}-(.5,0)_>>{#1}}
\def\Extend{*!C\txt{\bf\vdots}}

$$\xy\xygraph{!{<5pc,0pc>:<0pc,3.3pc>::@(}
   [] !{\OutNeuron a}
  [d] !{\OutNeuron b}
[llu] !{\InNeuron{x_1}}
  [d] !{\InNeuron{x_2}}
 [ruu]!{+(0,.25)}!{\Hidden}
```

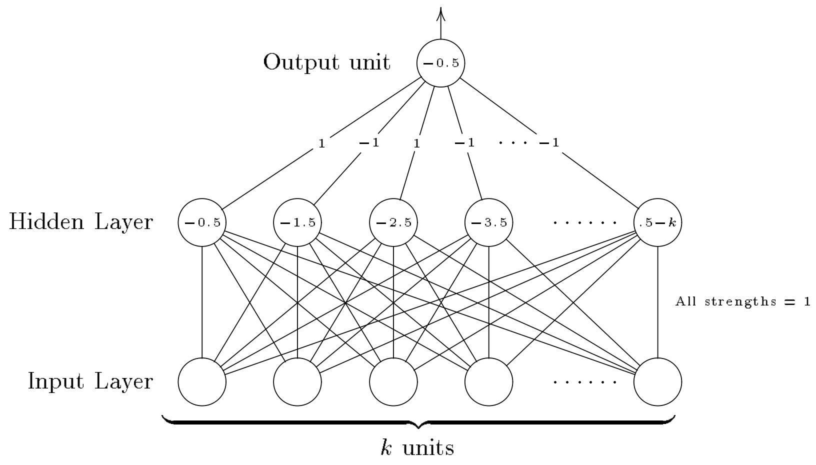


**Figure 5**: A network that computes parity.

```
 [d]!{\Hidden}
 [d]!{+(0,.25)}!{\Extend}!{+(0,.25)}
 [d]!{\Hidden}
 [d]!{\Hidden}
!{@i@)}}\endxy $$
```

The "hidden units" connect to all the "inputs" and to all the "outputs". This is achieved within the XY-pic diagram by using the "stack" construction. Initialisation of the new stack is done by `@(` while `@+`, used in `\InNeuron` and `\OutNeuron`, pushes positions onto the stack. Within `\Hidden` the `@@{...}` construction interprets the code within `{...}` for each position in the stack. Thus each call to `\Hidden` makes a solid line connection from its position to each input and output node. The second use of `@i@)` empties the stack before leaving it, when it is no longer needed.

Many of the above techniques were used to produce figure 5, which is based on the diagram on p.142 of [1].

```
\def\OutNeuron#1#2{\Neuron{#1}
 \POS="O"-(.5,0)*!R\txt{\small #2}
 \POS"O"\Output\POS"O"}
\def\Neuron#1{\POS*!C{#1}
 \POS*+++[o]{}*[o]\frm{o}}
\def\Output{\ar +(0,.35)}
\def\Input{\Neuron{}
 ;@@{\connect\dir{-}}\POS p}
\def\Hidden#1#2{\Neuron{#1}\POS @+
 \ar @{-}"O"|{#2}}
\def\Extend{\drop\txt{$\bf\cdots\cdots$}}
\def\Extendd{\POS="2"\Extend\POS
 ;"O"**\dir{}?*\txt{$\bf\cdots$}\POS"2"}
\def\longbrace#1{\POS"a"."b"
 !C.*++\frm{_\}}!D*++!U\txt{\small#1}}
\def\leftxt#1{\POS*++!L\txt{\tiny #1}}
\def\nudgeL{\POS-<.7pc,0pc>}

\def\objectstyle{\scriptscriptstyle}
\def\labelbox#1{\object+<1.5pt>
 {\scriptscriptstyle#1}}
$$\xy\xygraph{!{<3pc,0pc>:<0pc,5pc>::}
 []!{\OutNeuron{-0.5}{Output unit}}
[dlll]="H"
 !{\drop!R\txt{\small Hidden Layer}}
 !{+(.5,0) @i} !{\Hidden {-0.5}{1}}
[r]!{\Hidden {-1.5}{-1}}
[r]!{\Hidden {-2.5}{1 }}
[r]!{\Hidden {-3.5}{-1}} [r]!{\Extendd}
[r]!{\nudgeL\Hidden {.5-k}{-1}}
"H"[d]!{\drop!R\txt{\small Input Layer}}
 !{+(.50,0)}!{\Input}="a"
[r]!{\Input} [r]!{\Input}
[r]!{\Input} [r]!{\Extend}
[r]!{\nudgeL\Input}="b"
!{@i \longbrace{$k$ units}}
"b"!{+(0,.5)\leftxt{All strengths $=1$}}
}\endxy $$
```

**Overview**

A structured programming approach seems to be most appropriate for these Neural Network diagrams. The "top-level" specification should be easily human-readable, containing a specification of the existence of the various objects within the diagram, together with text for any associated labels. Also at this level is information concerning the visual layout. Typically the user invents control-sequence names (macros) for the objects; names which parallel the jargon commonly used in the particular field-of-study.

The "lowest level" of specification contains the details of how a particular (part of an) object is to be drawn on the page, using XY-pic kernel language or features. This allows easy access to the code to specify size, shape, exact spacing, placement of labels relative to the object being labelled, styles for links, whether or not arrow-heads are required, and a myriad of other details.

Between these levels are the definitions of the user's macros. The high-level objects generally will have some structure, some of which may be in common with other objects. Sometimes there will be a need to differentiate between subcases of a class of objects. This level tends to consist of definitions in terms of macros which are not used explicitly at the top level. Drawing an analogy with the neural net diagrams presented above, this level of code represents a "hidden" layer which interfaces the top-level code as "Input" to the XY-pic code that actually places marks on the paper as "Output".

Many types of diagram can be produced by making just small changes to code presented in the above examples. The XY-pic Reference Manual[2] is the primary source to peruse for the coding required to obtain a desired effect.

### References

[1] Igor Aleksander and Helen Morton. *An Introduction to Neural Computing*. Chapman and Hall, 1990.

[2] Kristoffer H. Rose and Ross Moore. XY-pic reference manual. Mathematics Report 94–155, MPCE, Macquarie University, NSW 2109, Australia, June 1994. For version 2.10. Latest version (3.0+) available by anonymous ftp in `ftp.diku.dk: /diku/users/kris/TeX/xyrefer.ps.gz` or in `ftp.mpce.mq.edu.au: pub/maths/TeX/xyrefer.ps.gz`.