# Neural Networks: *What's Inside?* The Explicitness Hypothesis.

*Jan Lemeire, jan.lemeire@vub.ac.be, 2001*

*- curiosity killed the cat -*

## Abstract

In this text we will investigate the inside of a neural network. This is against the widespread belief that we cannot understand a neural network. The information would be spread out throughout a neural network. I advance a hypothesis to attack this beliefs. I will try to proof that neural networks represent redundancy functions and concepts *explicitly*. The proof is elaborated for recognition problems and uses the properties of training, reuse and flexibility, which are considered to make up the power of neural networks. I consider this text as a start for an in-depth investigation that is worth while, because the consequences of the hypothesis are enormous.

## Introduction

How much research investigates the inner of a NN? Is there there a theory of how to build NN? How many nodes and layers should be used? So we don't ask how a NN will do the job? We just train it and the functionality will appear?
It is a widespread belief that it makes no sense to look inside a neural network (NN), that it is impossible to understand what each neuron represents. Knowledge is *implicit* in the network, it is spread throughout the network. I want to argument a hypothesis that goes against this *don't-even-try-paradigm*.

The explicitness hypothesis consists of 2 parts, explicitness of redundancy and of concepts. For the first part I use the term 'redundancy', which is unfortunately not the perfect term, but the only one I found so close to what I want to say.

The proof is developed for recognition problems and I use the recognition of figures for demonstration purpose.

## Example: Recognition of figures.

Consider the task of constructing a system that recognises figures in an image of m by m black or white bits, figures like rectangles, equilateral triangles, trapezoids, etc. These figures are defined with a limited number of concepts like number of angles, right angles, parallel sides, equal sides… They should be recognisable in any position or size.

## Definitions

These are not complete definitions of all terms, but only the necessary for understanding the proofs.

A **recognition problem** can be seen as the *construction* of a demanded subset of the state space, the **object** that has to be recognised is a subset of all possible input vectors. A node of a neural network 'represents' a **property** of the input, because the value of the node is activated for a subset of the state space. **Training** of the NN is defining the weight of every link with a limited number of instances of the state space. **Reuse** means reusing a node in the construction of new objects. **Flexibility** means adapting the NN to recognise a changed object. **Concepts** are defined as subsets of the state space that are used in the definitions of the objects. Like a square can be defined with the 3 concepts quadrangle, right angles and equal sides.

The hypothesis is an existence-proof, it shows that explicitness leads to efficient construction of the recognition system. But it doesn't say how to build it.

## Part 1: Explicitness of Redundancy

"If there is redundancy in the definition of an object, than this redundancy *must be* explicit in the NN if we want efficient training over this redundancy."

For example, the position of a figure is unimportant, the NN should recognise the figure at all positions. So there is redundancy among these translated instances. Figures can be freely translated, that's what I call a **redundancy transformation function**. So, redundancy can be defined by a function that tranforms any instance of the object into another instance.
Formal:

- For every instance a property r can be defined.
- There exists a function f so that for all instances of the object and for all values of r: f(instance, r) is also an instance of the object.

**Training over this redundancy** is defined as training the NN by instances that don't cover all redundancy. We want the NN to learn to recognise the figure at any position, while the training examples cover only some positions. So, in the example, we want the NN to learn the figures independently from their positions.

The theorem then postulates that the redundancy will be "removed" from the input so that the part of the NN that will learn the figure "sees" all instances as if they are at the same position. The NN is split into 3 parts (Figure 1), f "dissolves" the redundancy in order that for NN' all C & f(C, r) give the same input values in NN'. F thus transforms the input into redundant-free input for NN', so that for NN' all redundant instances look the same. R is the redundancy parameter.

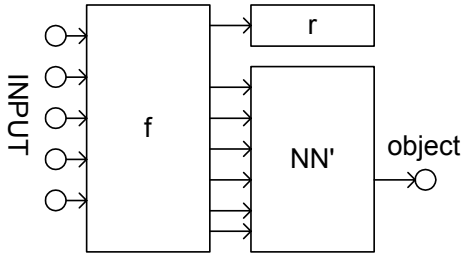


Figure 1: explicit redundancy

Now it can easily be seen that if NN' is trained with some instances and can classify these instances, all redundant instances will also be recognised by NN'. If there would not be such a part, redundant instances result in different node values and after training of the links for some instances, other redundant instances can't be recognised by the same weights! So, by further training with redundant instances, the originally trained weights are destroyed. A commonly accepted problem of NN training.

I don't claim it has to be like this, but it is just very simple and efficient! I also don't say how this f looks or how to construct it. I don't belief it is easy, maybe present NN can't implement this functionality. But we have to find a way to do this.

In the case of the figures, other redundancy is rotation, scale, the notion of a side, an angle, … So for construction of the NN we have to separate this redundancy from the definitions, what I call the **constraints**. Moreover, when a redundancy form is made explicit in a NN it can be applied for learning other objects leading to reuse of redundancy. Also for flexibility this approach is efficient. When changing a definition of an object, it will be applied over all redundancy. Even more surprising is how the human brain learns objects. The redundancy forms are not stated in the definitions, only its constraints. When learning the definition of a figure, we automatically know that we can translate, rotate or scale it and how we have to do this!

## Part 2: Explicitness of Concepts.

A concept C is **explicit** in a NN if there exists a node n so that n ⇔ C. This is in the network downwards. A node n is **discriminating** in the construction of an object o if o => n, so n must be true for recognition of o. This is in the network upwards. See Figure 2.

Hypothesis:

| If there exist concepts that are reused and must be flexible in object definitions, then the constructed NN with these concepts explicit and discriminating is the minimal training solution. |
|---|

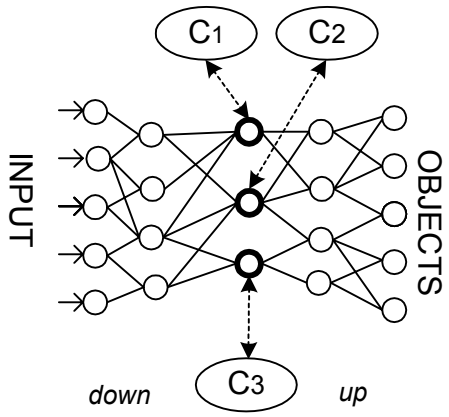


Figure 2: Explicit Concepts.

So a NN where the concepts of straight angle, parallel sides, triangle, etc. are explicit will be the most efficient construction for learning figures. Moreover, we could also easy learn the system that a square is a rectangle with equal sides.

The proof consists of two parts, I have to proof for the upper and lower part of the NN that they are more efficient than any other NN. The proof for the upper part is easy. For construction of a new object, the NN can simply reuse the nodes representing the necessary concepts (Figure 3). For flexibility it is only necessary to change the weight of the link to that node, because it is discriminating (Figure 4). This can't be done in a more efficient way.

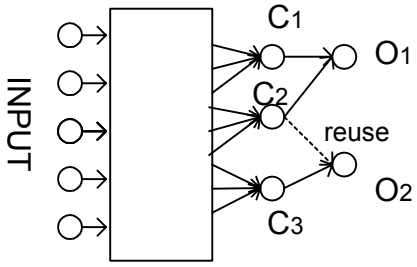


Figure 3: Reuse.

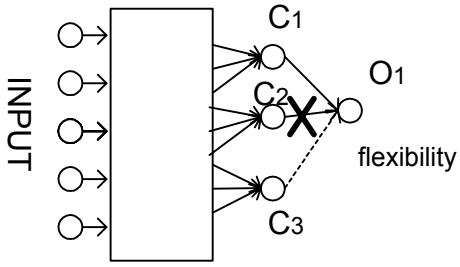


Figure 4: Flexibility.

To proof that the lower part is the simplest solution is not so easy. I will try to do this by representing the NN as set calculus. Figure 5 shows how a node/object is defined as a sum of other nodes/concepts. Figure 6 shows how reuse and flexibility is performed.

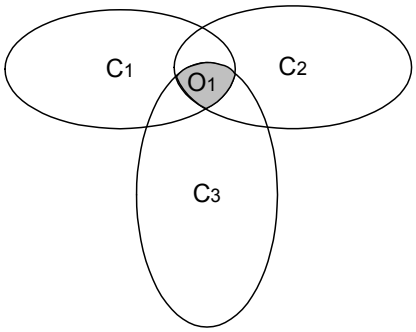


Figure 5: Set mathematics.

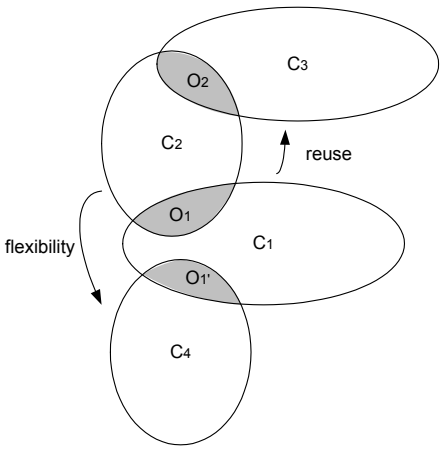


Figure 6: Reuse and flexibility with sets.

In a counterexample, the object set should be constructed in a different way, with different sets. So I should proof that these sets are more difficult to construct than sets representing the concepts. To proof this formally a construction proof is necessary, which I don't have for the moment. But I can give an intuitive proof which gives a good indication of directions for further investigation and a possible link with the explicit redundancy theorem.
We should look at what I call the **discrimination** of the subsets (Figure 7). In the construction of object O1, it is the concept C2 that serves for discrimination d1. In the construction of O2, we reuse C2 for discrimination d2. Now it should be proved that reusing C2 is simpler than doing d2 by another set, that d1 & d2 are the *same* discriminations (if you can do it for d1, you can do it for d2), *independent* from other sets (here C1 and C3). So C2 should be a recognition problem independent form C1 and C3. We can understand this, recognising a right angle is independent whether it is a triangle or a quadruple. For flexibility we come to the same conclusion. Changing from O1 to O1' by changing C2 into C4 is the simplest when constructing C2 and C4 than either changing (C2 ∩ C1) into (C4 ∩ C1).

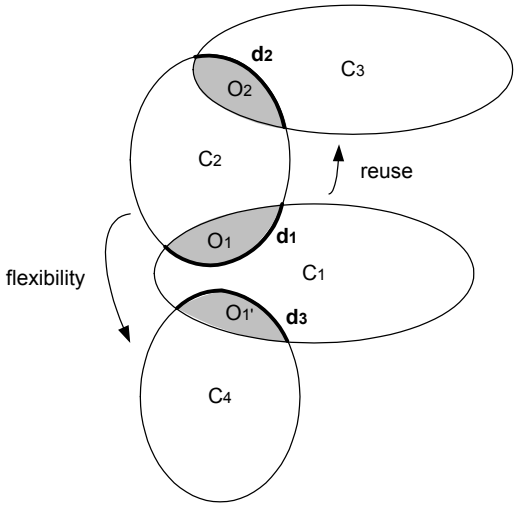


Figure 7: discriminating subsets.

In the figure example the concepts are known, in OCR for example not. But the hypothesis claim that they will become explicit in an efficient NN. Of course we could define *any* complex concepts so that this hypothesis doesn't hold. A construction theorem should make clear that these concepts 'naturally' appear, independently and objectively.

## Future Work

- How to seperate the redundancy?

- Formalisation of the redundancy proof for counterexamples.
- How to construct the concepts?
- Last part of the explicit concept proof. By combining it with the explicit redundancy? Concepts are constructed by redundancy and constraints?
- Can this hypothesis be enlarged for other than recognition problems?

## Conclusion

By proving the necessity for explicitness in neural networks, I make a start for a construction theorem. Because the consequences of my hypotheses are substantial:

- There is a structure!
- We have to analyse the redundancy and the concepts.
- A neural network is not a magic bullet, there is a best solution.

The proof is based on the properties reuse and flexibility, which are of course not necessary in *all* problems, but they create structured and efficient systems. Besides, we find these properties for a full 100 % in the human brain and explicit construction sounds very 'natural' to me.